Teamwork in Architectural Modelling: Representation and Communication Requirements for Computer Support in Collaborative Design

Chengzhi Peng

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy
to the
University of Edinburgh
1994

To My Parents

Abstract

This thesis presents a study of what is required for a computer system to support human creative collaborative design activity. The field of design study is targeted on architecture. Design is seen essentially as an activity of modelling complex objects with or without computers. "Teamwork in architectural modelling" is, therefore, the realm of inquiry, with the goal of identifying the representation and communication requirements for computer support. A phenomenon of collaborative design to be explained is how the distribution of design work among multiple participating expertise is related to the integration of individual design contributions into products with architectural integrity.

Several historical cases of building design projects are studied. We examine the design expressions produced by members of the design team in each case. Based on the results of case studies, we put forward two distinct teamwork patterns in collaborative design: metaphorist and structuralist. The teamwork patterns are subsequently treated with more elaborate analyses and simulations. Prior to the presentation of the requirement studies, a survey of the current state of the art in building collaborative drawing and design support systems is provided. The survey shows that our concern of supporting collaborative design modelling remains a subject largely unexplored by the current system designs.

For each teamwork pattern identified, a situation-theoretical framework is adopted for carrying out a more elaborate analysis. Designers' performing of modelling acts in individual and group modelling spaces gives rise to a number of situation types. By examining the conditions for design information to flow among the situation types, we arrive at constraints on collaboration. Representation and communication requirements for system support are then discussed in accordance with the constraints identified. Given the metaphorist and structuralist requirements studied, we continue to explore two potential forms of collaborative design computing by simulating some simple design examples in the algebraic specification language OBJ3. The simulations suggest a number of more specific issues in designing components of computer support. Finally, by commenting on the current findings, a direction for further research is proposed.

Declaration

This thesis has been composed by myself and it has not been submitted in any previous application for a degree. The work reported within was executed by myself, unless otherwise stated.

Chengzhi Peng

April 1994

Acknowledgements

I wish to thank the joint guidance given by my supervisors Dr. John Lee and Mr. Aart Bijl. Over years, they have shown me their constant encouragement and, above all, considerable patience in my development of the thesis. It is with the help of their often stimulating questions and constructive comments that my interest and effort in pursuing this work has constantly been kept alive.

Researchers at EdCAAD have maintained a rich research environment. Among them, Dejuan Wang taught me how to programme in OBJ3. Discussions with members of the design discussion group, including Peter Szalapaj, Stavros Vergopoulos, and Satoshi Imamura have helped to shape the theme of the thesis. Thanks also to the constant concern from Margaret McDougall, and the essential system support from Ian Thurlbeck.

It would have been impossible for me to carry out the survey of collaborative drawing tools, if help was not given by the researchers I have contacted with. I wish to thank Sara Bly, Tom Brinck, Gerhard Fischer, Saul Greenberg, Hiroshi Ishii, Fred Lakin, Jeff Lee, Iva Lu, Marilyn Mantei, John Tang, and Catherine Wolf for providing me with essential research papers, technical reports, and numerous e-mail discussions. Regarding my submission of the survey paper to the journal of Computer Supported Cooperative Work, I am grateful to the anonymous referees' positive questions and comments upon which my revision of the survey was based.

At a later stage, I was extremely fortunate to be given the opportunity to join the CSCD project at ITRI, University of Brighton. I wish to thank the support and encouragement from Dr. Lyn Pemberton, Dr. John Downie, and Professor Donia Scott. With their help, I have been able to work continuously toward the end of the thesis.

As an overseas research student, I have benefited financially from the support given by the CVCP Overseas Research Student Award Scheme during 1990/91 and 1991/92. Finally, I also wish to express my gratitude to the Participatory Design Conference 1992 Committee for their awarding me an assistant scholarship, which enabled my trip to that thoughtfully organised conference at MIT in November 1992.

Contents

List of Figures

List of Tables

Chapter 1

Understanding and Supporting Teamwork in Design

> *It is impossible to divorce the building from its legal, technical, political, and economic context. At the same time, a major part of any design approach is the way constraints may be absorbed and whenever possible inverted into positive elements.*
>
> — *Sir Richard Rogers, 1980*

As an introduction, this chapter is intended to give an overview of the subject matter, problem context, research objective and method that constitute this thesis on the requirements for supporting creative human collaboration in design. Rather coincidentally, this thesis was developed at the time when computer-supported collaborative design (CSCD) has become a distinctive research area. CSCD now attracts many researchers to work on experimental tools and systems from various perspectives, including, to name some of the most active, Distributed Artificial Intelligence (DAI), Enterprise Integration (EI), Concurrent Engineering (CE), and Computer-Supported Cooperative Work (CSCW). As an alternative perspective on what computing methods and design environments can be developed to assist or participate in collaborative design activity, this thesis adopts a research perspective pertaining to *Design Studies*. The field of design concerned is architecture or, more broadly, built environment. It is believed that an enquiry into the nature of human collaboration in designing buildings can open up further fundamental questions of computer-supported collaborative design which may in turn motivate new ideas of system design.

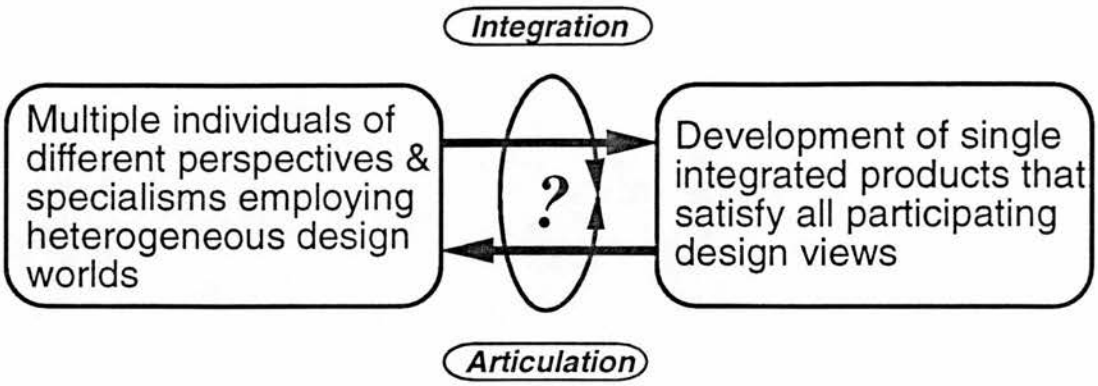


Figure 1.1. The basic issues of collaborative design set out by a view of group practice in architectural design.

1.1 Basic Issues Raised by Architecture

There are two main reasons why it is thought in this thesis that an investigation into *teamwork in architectural design* can contribute to an understanding of collaboration among people when designing things in general: Firstly, as a matter of technical necessity, the design of buildings, or, more broadly, of environmental artefacts, has to be based upon the participation of multiple design experts, each of which is more capable than the others of dealing with a particular design domain of a project; secondly, as a consequence of exchanging critical judgements among participants in the course of designing, the production of final unity in design products is a common concern shared by all parties of a design team.

To put the above perception of the problem context more clearly, let us consider a picture of design practice in focus. In the world of architectural design, we often see two salient features in the entire practice of design production, which may be better illustrated by the diagram shown in Figure 1.1. On the left hand side, there is the participation of multiple individuals of various design perspectives and specialisms, who may act on heterogeneous design worlds to undertake domain-specific design tasks. On the other hand, there is the goal of developing a single integrated design of an artefact that satisfies all participants' design judgements. This overall picture of group practice in design suggests three basic issues of collaborative design which may be better understood:

Seen from left to right, the first issue is about teamwork in *design integration*. The

second issue, seen from right to left, is to do with teamwork in *design articulation.* And thirdly, being, perhaps, less straightforward than, but closely related to, the previous two issues, there is the interplay between integration and articulation. It can be argued that any model or theory proposed to explain what is involved in collaborative design must encompass these three aspects. This thesis aims to propose such a descriptive theory and show its implications for the development of computer support.

1.2 Putting CAAD into Group Practice in Design

There are always many designers working together in producing meaningful and useful environmental artefacts. Though design practice is traditionally teamwork-based, modern designs and implementations of computer-aided architectural design (CAAD) systems have not entered the realm of supporting group design activities. We have not seen systems developed deliberately from the pespective of supporting direct or indirect communication and coordination among members of design teams for the purposes of integrating and articulating individual contributions.

As we can see in the 60's and 70's, some public institutions employed computer system developers to build up integrated CAAD systems with the intention of embracing nearly all aspects of building design and production. Classical examples are the OXSYS system [Hos77], developed for hospital design, and the SSHA system developed for low-cost housing design [BSR79]. Notably, these early pioneering integrated CAAD systems were developed under two conditions: firstly, these integrated design systems were targeted at particular regularised building construction methods (e.g., the Oxford Method of construction in the OXSYS system, and the Scottish Special Housing Association Standards in the SSHA system); secondly, what contemporary computing systems could provide were highly centralised services [Bij89], which did not facilitate the more modern notions of distributed operations.

When less centralised and lower-cost computing resources were made available by the information technology advanced in the 80's, knowledge-based approaches to CAAD system engineering came into place. The new trend adopted novel techniques of interfacing or networking several diagnostic or generative Expert Systems as the features of integrated design environments [Pen89]. Examples from the work done at Carnegie Mellon University were the use of knowledge-based programming techniques for interfacing knowledge-based systems to database management systems (KADBASE) [Reh85], and the Integrated Building Design Environment (IBDE) [Sch90]. Clearly, these computing approaches aimed to automate some portions of the design tasks by compiling formal

design evaluation or generation procedures into specialised knowledge bases interlinked via a blackboard architecture.

It is evident that those early integrated CAAD systems and the later knowledge-based design environments present a rather optimistic assumption: under the command of some solitary individual, design can be achieved by machine processing of large established bodies of design data or knowledge.

Within the research field of design studies, a number of interpretative frameworks describing the social and distributed nature of design activities were proposed (see, for instance, [Law80, War87, Buc88, HG88]). Apparently, these descriptions put a greater emphasis on 'group dynamics' instead of on what specific knowledge of design is involved. However, the studies of participation and group interaction in design have not systematically resulted in requirements for the development of design computing environments that support teamwork.

Rather recently, the problems of what and how to develop communicating and computing systems that can be supportive for people involved in *collaborative design* have been researched not in the field of CAD but in Computer-Supported Cooperative Work (CSCW)[1].

In particular, as a sub-area of CSCW research, a large portion of the research on *shared workspaces* has to do with building prototypes of *shared drawing space*[2]. Being motivated and guided by earlier observational studies of working group graphics and shared drawing space activities, researchers have been implementing a range of prototypes of *collaborative drawing systems.* Being different from traditional drawing systems, the CSCW-oriented drawing tools are primarily devised to be used by a group of users for undertaking experimental *collaborative design* in various spatio-temporal situations.

1.3 Design as Modelling Complex Objects

In investigating computer-supported collaborative design, a basic position is taken throughout the thesis: design is viewed as an activity of modelling complex objects.

[1]The term "computer-supported cooperative work (CSCW)" was first coined by Irene Greif and Paul Cashman in 1984, calling for researchers and developers to participate in a series of ACM-sponsored conferences that examined how people work together and how information technology may support them. The conferences led to the subsequent publication of the first book on CSCW, advocating CSCW as a distinctive research field [Ge88].

[2]This conclusion is drawn from Saul Greenberg's well annotated bibliography of CSCW papers published during 1986-1991 [Gre91].

Design is such a rich activity that it is not possible to characterise the nature of design via a single particularly adopted perspective, for instance, information gathering or processing, representation, actions of making, evaluation etc. None of them can be said to be more correct or important than the others. It is merely our intuition that the current view of design as modelling complex objects is, relatively, a more comprehensive one as explained in more detail below.

Owing to the large varieties of components as well as the dynamic relations between parts and whole, the objects designed by architects or engineers tend to be complex in nature. However, there have been alternative techniques devised to convey and manage the complexities involved in design. As can be observed in many architectural or engineering workshops, designers often use and manipulate physical tokens (e.g., cardboard, strings, paper, wooden blocks and polystyrene, etc.) when they construct models for various purposes. Working with these physical models, designers are better equipped to develop, reflect, and communicate design ideas with themselves and others. Though it can be extremely tedious, model making has been generally considered by design educators and practitioners as an essential part of design processes. The construction and use of physical models has been widely observed not only in individual cases (see, e.g., [Jan78, Gol88]) but also in group processes (see, e.g., [Sch85, War87]).

In addition to making physical models, designers always produce drawings. There are intimate relations between the production of drawings and model making. Sometimes models are made after drawings have been produced (i.e., drawings serve as blueprints in model construction); and at other times, drawings are produced on the basis of models constructed (i.e., models serve as referents or primary sources for drawing construction). In, perhaps, most cases, designers work in quite a mixed manner; that is, designers produce drawings to develop or elaborate design solutions suggested during model construction, and designers construct models to better inform themselves of the consequences of associating or disassociating design ideas explored in drawings. It is therefore reasonable to say that, in designing complex objects, there is a need for constant switching between drawing making and model making. Seen in this context, drawing can be thought as a somewhat abstract form of design modelling.

As already pointed out by Tjalve and his colleagues in their systematic introduction of 'engineering graphic modelling' [TAS79], a drawing is a model if it is made to demonstrate the following attributes:

- A drawing represents *modelled properties* (e.g., structure, form, material, dimension, surface, etc.);
- A drawing has a *receiver* who is the person or persons to whom the drawing communicates information;
- A drawing is *coded* in systems of symbols (e.g., coordinates, graphical symbols, types of projection) known to the receiver.

Seen from the viewpoint of design as modelling complex objects, the activity of designing in general can be said to encompass two interrelated aspects: the representation of conceptual structures and the performance of modelling actions. The former decides to a large extent the range of basic construction elements and their properties for design use; the latter, when performed by individuals, can lead to specific design descriptions (depictions).

Design as modelling is even more meaningful and useful when considering the development and use of computers as design tools. The kinds of conceptual structures, as largely embedded in the processes of making physical models or drawings, can be made overt when the models are constructed in an electronic design studio. Given the advancement of computer science and technology, modern CAD systems have provided designers with various computational devices to do so. One of the main benefits of doing so is that a dynamic integration of model construction and design production can be achieved. A good example for illustrating this rather hard to achieve but more promising CAD approach can be found in the MOLE project [Kri85, Bij87, TB88, Whi92], where a general descriptive formalism, "kinds-slots-fillers", is provided for designers to represent and evolve their own conceptual structures that can then be used by themselves to instantiate specific design instances.

There are researchers who advocate the importance of *activities* that are observed in collaborative design sessions (e.g., [Bly88, Tan89]). To meet the requirements for direct communication among collaborators, the supply of real-time supports for group interaction in design (i.e. via talking, gesturing, sketching and writing etc.) among geographically distributed participants is of primary concern. Given those CSCW findings and experiences, this thesis proposes that *design as modelling* can be an alternative perspective for investigating the communicative aspects of collaborative design. The objective of the investigation is to identify conceptual frameworks that unify the aspect of activities and that of artefacts, such that the basic criteria for defining CAD tools with CSCW features can be articulated.

1.4 Exploratory Collaborative Design Computing

It seems that we now have two major resources for research into computer-supported collaborative design: one is from *collaborative computing* which can now be achieved through rapid combination of various computing and communication technologies, and the other is from *design computing* which already presents diverse approaches to the making of design tools and environments in CAD. As pointed out by Grudin [Gru91], collaborative computing utilises networking, communications, concurrent processing, and windowing environments. These new techniques and facilities have opened up the possibility of collaboration via group interaction in real-time even if participants are geographically remote to each other. To better inform ourselves about the state of the art in collaborative computing, especially in the latest designs and implementations of group drawing tools, we dedicate a whole chapter to a survey of this exciting development.

Working closer to some domains of design practice, some CSCW researchers already considered that the sharing and use of certain kinds of structures of *design artefacts* by group members can be captured in a formal language or a dedicated design system (e.g., [Lak90, FGL+92]). Collaboration-supporting design tools developed in this view contain an underlying assumption that the structure and knowledge of the design products built into the systems is stable and usable to most design practitioners over the lifetime of the systems.

Research in theories of design computing has shown a different approach to that of knowledge-based one. Again, referring to the MOLE experiments, it has been shown that designers can work with a general descriptive system in constructing their own models of design objects in pictures and words. As proposed by Bijl [Bij86] and [Bij89, pp.172–208], a more fundamental position is to allow users to be able to instruct machines what to do in a CAD modelling environment with minimal or no prescribed domain-specific knowledge.

In drawing possible links between the existing developments in collaborative computing and in (non-prescriptive) design computing, a theme of study into computer-supported collaborative design is emerging: *exploratory collaborative design computing.* Given the background of enquiry, this thesis work sets out to investigate the requirements for supporting design as an explorative process involving communication and coordination among a team of designers. It is expected that the requirements derived from the present study can serve as a useful set of pointers to future development of a collaborative design computing framework.

1.5 An Approach to Requirements Study

Having introduced the background and the scope of study, the next ingredient of the thesis to be introduced is, naturally, the research method. Given the aim of research as indicated above, we have chosen to perform a requirements study. In the field of software engineering, the importance of formal or informal approaches to capturing and specifying system requirements has been recognised both in theoretical development and in industrial practice [FKN+92, WL88]. At least two benefits of addressing requirements specification or engineering can be mentioned. First, good specifications may save developers from making mistakes at an early stage that often result in a huge increase of system development cost [PST91]. Secondly, formal specification of a computational architecture can be an economic and sufficient way of expressing one's understanding of the functionality of the system without actually building it [Cra91].

The requirements study carried out in this thesis, however, did not reach (nor was intended to achieve) a formal specification of a system architecture at the level of mathematical precision. One obvious excuse is the limited time and space allowed for a thesis of a reasonable length; the other reason is due to the current rather poor understanding of the subject matter which by no means can be handled neatly by a complete and sound specification. Instead, along with the development of the thesis over years, our approach to the requirements study consists of three parts: case studies, requirements classification, and concepts modelling, each of which deserves more discussion.

1.5.1 Observing historical cases

A number of case studies were carried out at the outset. Three examples of teamwork in design are eventually presented in the thesis. These cases examined are 'historical' in the sense that they were selected from existing design documents or research literature that gives reports of *real* building projects undertaken by some design teams. The data collected are mainly instances of diagrams, drawings, illustrations of models and the designers' or researchers' verbal descriptions annotating the graphical evidences.

Case study or *protocol analysis* has been widely practised in conducting research into design thinking, design teaching and learning etc. (See [Row87, Buc88, Sch91] among many others). In a more 'laboratory-based' case study, with modern means like video cameras, tape recorders, a huge amount of data can be captured for later analysis, including gestures and even facial expressions. Subjects are invited to carry out some given design problems or tasks in rooms equipped with data-capturing facilities. The drawings and other design expressions produced by the subjects are the outcome of

some, at most, hours of work.

Case study based on historical material has a different flavour. Without the kinds of artificial controls imposed by laboratory-based studies, the design representations and expressions presented in historical cases are often the outcome of much longer design processes (involving many people over years not hours). Since, from a research point of view, making sense out of what designers produce is an important part of enquiry, we should be aware of the fact that, in relation to time-limit (i.e., hours or years) and the initial working condition (i.e., being in simulation or being real), designers may produce things of different natures that may reveal what design is about very differently both in individual as well as teamwork cases.

The above is not to belittle any of the methods discussed but to draw out limitations of case studies. It is certainly interesting to compare the understandings of design resultant from different attitudes toward protocol analysis in design cases. Nevertheless, as far as case study is concerned, one thing is clear: The full spectrum of teamwork in design is by no means represented by these three examples presented here, but they do provide sufficient material and variation to serve as an informative backdrop for later conceptual expositions.

1.5.2 Classifying teamwork patterns

The basic unity of each of our case studies is the design project. Within a design project there are participants working in various domains of the project and a collection of graphical expressions which are viewed as 'traces' left by some members of the design team. Two kinds of reading are followed: direct reading of the depictions and a reading of the textual descriptions associated with the depictions. The readings lead to a distinction between two basic categories of design information: commonly shared and individually held. Given the boundaries initially drawn, we then ask what are the possible relations between the two categories of information (i.e., between the shared and the individual). It turns out that there appear two distinct teamwork patterns in our current case studies: metaphorist and structuralist, discussed in Chapters 4 and 6 respectively.

In our attempt to give a deeper analysis of the teamwork patterns differentiated, we apply a theory of information, *situation theory*, which is concerned with the "flow of information". Here we make a basic assumption, that is, collaboration among members of a design team can be characterised by explaining how information flows across the boundaries of different types of representation.

After an overall picture of the flow of information is gained, according to situation

theory, we are in a position to spell out the *constraints* on information flow. Following our line of enquiry, this is equivalent to a spelling out of the *constraints on collaboration.* Guided by the constraints identified, a final part of the conceptual exposition is to produce a classification of representation and communication requirements for both the metaphorist and the structuralist approach.

1.5.3 Modelling the components of collaborative design computing

The third part of our requirements study is concerned with a more fine-grained examination of the constituent concepts of teamwork in design as readily proposed by the preceding analyses. This part of work involves computational modelling of the framework of design sharing and distribution operated in each teamwork pattern. The study is carried out more like performing some 'design exercises.' These exercises were also intended as an exploration into two potential forms of collaborative design computing: *joint substantiating of common generic structures* and *joint abstraction of shared integration schemas.* We firstly give examples of design constructs and instances as contributed by some imagined design participants working as a whole or as markedly different individuals. Given the representation context set up in the first instance, we then examine the communication and coordination issues that may be necessarily involved in design integration and articulation.

The modelling environment used to carry out these exercises is provided by the functional programming language OBJ3, designed and implemented by Goguen and Winkler at the SRI International [GW88]. As an experimental language, OBJ3 presents several features that are not common to other high-level programming languages. These include the mechanisms supporting hierarchical *modulisation* and sophisticated *parameterisation.* Both features were extensively utilised in the construction of the representation contexts that simulate the exchanges of information among different domains of design representation as seen in the metaphorist and structuralist patterns. Our experience of using OBJ3 shows that the language itself is relatively easy to work with, and the exercises done in this manner demonstrate some experiences of requirements study concerning a combination of informal conceptual analyses with formal computational syntheses.

More specifically, with hindsight, it seems that the usefulness exhibited by the OBJ3 exercises is twofold: first, it evaluates our conceptual accounts by implementing, as we see it, some of the salient concepts of collaborative design in computing terms; second, it delivers a sharper view of supporting communication and coordination in collaborative design. The requirements study presented in this thesis suggests that the

most interesting and challenging functionality to be developed in a teamwork design environment is the capability of supporting *emergent communication frameworks*; we envisage a system which is able to accept and maintain the communication strategies and processes developed and evolved by participants in the course of designing.

1.6 Organisation of the Thesis

Without a division into parts, the remainder of the thesis consists of seven chapters organised in the following sequence. In Chapter 2, three historical cases based on three building projects are studied, each of which provides a rich collection of graphical illustrations of design objects. The graphical evidence we examined reveals, mainly, the representational bases for cooperation. In exploring further the interconnection between different types (or domains) of representation, we found it useful to characterise teamwork in terms of the interaction between *common images* and *distributed design developments*. In consequence, two generic patterns of teamwork in design are identified, providing two alternative conceptual frameworks for more elaborate analyses in later chapters.

Before a closer look at the teamwork patterns characterised in the previous chapter, we report, in Chapter 3, a survey of the current experimentation in building prototypes of *collaborative drawing support* systems. The survey has a particular objective to review how the researchers and developers of the prototypes have addressed the issues of supporting collaborative design. The review is organised into three aspects: (a) the key collaborative drawing activities that the tools were designed to support, (b) a framework for classifying the system design issues that the tools address, and (c) a catalogue of the system features of the various tools developed. The current survey shows that there are presently at least three different strategies of developing collaborative drawing support tools, which reflect the existence of diversified understanding and technological responses to what and how human collaboration in design may be supported.

To continue our enquiry into the patterns of teamwork in design initially identified in Chapter 2, an exposition of the 'metaphorist' approach to collaborative design is presented in Chapter 4. The focus is to explain how the production of integrated design is interrelated with the communication and coordination among participants of different design expertise. Our analysis begins with an abstract scenario describing the key elements that constitute the metaphorist pattern. We then examine the flow of information among different types of representation. Constraints on collaboration are

discussed in terms of the conditions of information flow identified. The metaphorist approach shows that *common design metaphors* which emerge from group processes can function as a communicative device, allowing for participants to collaborate effectively. We conclude some supporting issues regarding how to support the continual interaction between the inputs from local design decisions to the production of integrated design and the feedback from resultant integration to the individual courses of design developments.

In Chapter 5, following some of the key concepts provided by the metaphorist framework, a design exercise is set up to show how individual members of a design team can use their own concepts and methods for producing design expressions, and how those expressions can be correlated and integrated with those composed by other designers using different means. The simulation exercise is modelled computationally in the OBJ3 language; it shows that if joint abstraction of shared integration schemas is practised as another approach to collaborative design computing, there arise three supporting issues to be dealt with in a system design: (a) co-specification of spatial operations, (b) joint provision of source design expressions, and (c) coordination in making design changes.

As a companion chapter to Chapter 4, Chapter 6 is devoted to an analysis of the 'structuralist' approach to collaborative design. The aim of the chapter is to explain how design participants collaborate to achieve integrated design on the basis of sharing and substantiating common generic structures with domain design developments. As before, the analysis begins with an abstract scenario describing the key elements that constitute the structuralist pattern. The elements are then grouped into the aspects of model construction, model-constructing constraints, and modelling acts. The structuralist logic of collaborative design is found in the necessity of maintaining a dual correspondence between the evolution of *common generic structures* and the development of *domain design solutions* distributed over several sites. Following the constraints derived, the structuralist requirements for collaborative design computing are presented.

Like the simulation presented in Chapter 5, our second symbolic simulation of some design examples that bear some of the structuralist features is reported in Chapter 7. Two design examples are introduced to illustrate, first, a general practice of design substantiation—a process of giving more specific substances or contents to a generic design outline set up in the first instance. The second example, as an extension to the first, depicts a case of multi-party design substantiation. The main purpose of the simulation is to demonstrate what we mean by 'sharing common generic structures'

in computing terms. Especially, the formal supports for 'parameterised programming', including *theories, parameterised modules, actual parameters*, and *views*, provided by the language OBJ3, are used extensively in the simulation. As far as a system development strategy is concerned, our current simulation suggests that a basic theory of design substantiation and a parameterised joint-substantiation function may have an important part to play in developing a supporting environment for the structuralist approach.

Finally, in the concluding chapter, we present a review of what we have been through in the thesis. The background theory of the thesis is built upon our observations of the historical cases, in which we envisage the metaphorist and structuralist patterns of collaborative design. The focus theory is presented in the subsequent situation-theoretical analyses of the two teamwork patterns identified earlier. And the OBJ3 simulations of some collaborative design examples provide, we think, the data theory of the thesis. Limitations of our current work are discussed. To show that the finding of the thesis, in particular, the differentiation between the metaphorist and the structuralist patterns, is not an isolated one, we look at some related findings in other areas. The chapter ends with a proposal of a direction for future research.

Published papers of the thesis. A record of the published papers that are closely related to the development and presentation of this thesis is provided in Appendix E. In respect of the thesis chapters, Chapter 2 was first presented in *Participatory Design Conference 1992*, and a more elaborate version of the paper was accepted for publication in the journal *Design Studies*. The survey of collaborative drawing support tools in Chapter 3 was accepted for publication in the journal *Computer Supported Cooperative Work*. A position paper abstracted from Chapter 4 on the metaphorist pattern was accepted for the AAAI 1993 Fall Symposium on *Human-Computer Collaboration: Reconciling Theory, Synthesizing Practice*. An algebraic specification simulation of joint shape construction in Chapter 5 was first published in *Edinburgh Architectural Research*, and was subsequently accepted for a poster presentation at HCI'92[3]. An earlier version of Chapter 6 on the structuralist pattern was presented in the AAAI 1993 Workshop on *Artificial Intelligence in Collaborative Design*.

[3]Due to the lack of financial support at the time of submission, a poster presentation based on the paper was not finally implemented.

Summary

This introduction has outlined the background of the thesis. It is argued that research into the issues raised by group practice in architectural design can make a general contribution to the study of computer-supported collaborative design. The issues cover the basics of group processes, including design integration, design articulation, and the interplay between the two. A quick look-back at the CAD developments in architecture shows that the goal of computers as mediums or tools to support designers working together as design teams still presents an original challenge.

In investigating architectural design as teamwork activities, a research perspective is adopted that views design as the activity of modelling complex objects. This view assumes that designers not only draw or make things directly but also describe and represent underlying conceptual or procedural structures or knowledge during designing. In this regard, computers can be best envisaged as design modelling tools. Having set out the background and perspective of the study, we project the research interest in exploratory collaborative design computing that places more weight on human designers' active uses of computers in establishing their own working relations. Given this research programme, the components of our approach to requirements study are explained, leading to the present structure of the thesis.

Chapter 2

Designers in Teamwork

> *A true team between architect, painter and sculptor, aiming at an organic synthesis of their work by symbolic association, needs an intensive exchange by its members. For only a thorough, mutual penetration into the ideas of the various team-mates may lead to that kinship of spirit which can make a creative entity out of the individual contributions. ··· If contributions of sculptors and painters are desired by a client, he should let the architect choose congenial collaborators at the initial designing stage of a building. Only such foresight will enable the architect to build up his team in time to let each member take part in the creative process of the basic space-conception, a process so essential to the result of final unity in the building.*
>
> — *Walter Gropius, 1961*

This chapter reports an observation of communication in collaborative design by examining several historical cases of building design projects. It will present three cases of teamwork in architectural modelling. Each case provides a collection of drawings or illustrations that depict the artefacts being designed and constructed. Looking at the cases from the design as modelling perspective, two important factors of collaboration are observed: (1) the existence of multiple heterogeneous worlds of representation and action that members of a design team work with, and (2) the emergent correlation between common goals shared by all participants and domain-oriented goals pursued by individuals. Starting with the notions of *common images* and *domain design developments* derived from the case studies, we put forward explanations of the aspects of group dynamics in collaborative design. The current exposition concludes two generic patterns of teamwork in design, which suggest a spectrum of possibilities in developing communication and coordination supportive architecture.

2.1 Searching for the Evidence of Teamwork

Like many other human activities, the design and construction of built environments has always involved many designers working jointly as teams. One obvious reason is that buildings as design objects are often too complex to handle very well in every aspect by a single person. It has been like this in the history of design as described by Khan [Kah35], and it is even more the case in modern design practice as reported by Middleton [Mid67]. Though we often see famous buildings being associated with famous architects, in reality, they are the outcome of teamwork over, more often than not, years of collaboration among partnerships or associates of various design specialisms.

Certainly, the phenomenon of group approach to design is not an entirely new subject to the research community of *Design Studies*. For this subject matter, a variety of interpretative frameworks have been proposed. One widely taken metaphor of group design process is, perhaps, design as *game*. Lawson reviewed several design games that were specially devised to model group dynamics in architectural and urban design [Law80]. Working closely with the game metaphor, Habraken and Gross constructed a computer program called "concept design game" that can record and then replay sessions of more than two participants' playing for control distribution and territorial organisation [HG88].

Protocol analysis and case study are other popular methods used by design researchers. We mention two examples. Based on his analysis of a documented design dialogue between an architecture student and a studio master, Schön proposed a theory of "reflection-in-action", acting as an epistemological framework of design learning and teaching [Sch85, Sch91]. A studio-based empirical study of a design project participated in by a group of architecture students was carried out by Ward [War87], in which seven subjects went through group processes and developed *archetypes* for a commercial complex project by, mainly, gathering together individually made cardboard models.

Rather recently, understanding and supporting collaborative design has become a prominent topic among some computer scientists working in the newly developed research field of Computer-Supported Cooperative Work (CSCW). As the amount of this research work has now reached a significant level, a comprehensive survey of the CSCW researchers' work will be given in the next chapter. Being different from the design theorists, the CSCW people look at group design activity with an eye of identifying specific opportunities for developing collaborative drawing or design tools. Also, the domains observed can be very different from that of architectural design, including, for instance, computer interface design, robot design, or group writing etc.

Given the orientation above, our searching for evidence of teamwork in design, through a number of case studies, is set out in the domain of architecture with the intention of identifying the group dynamics in collaborative design. As will be shown in later chapters, the dynamism of group interaction identified spell out important requirements for building computer-based group design tools.

Still a few more words need to be said about the nature of our case studies. In contrast to the design studies mentioned above, we take a different measurement. It is considered that an investigation of cooperative design can be approached by analysing, not simulated nor controlled, but naturally created and evolved design expressions taken from historical cases of building projects. We have explained the basic rationale of doing this in Chapter 1. As will be shown at the end of this report, the current study has delivered some results not seen in other research literature.

First, in each case, we see different kinds of design expressions that correspond to the participation of multiple individual design worlds and the production of integrated designs. Second, there are relations embedded among the different types of design representation; it is by explaining the relations that we arrive at some general conceptual accounts of group interaction in design modelling. What appeals to us in these concepts is that they are not bound to particular structures of design products or processes nor to specific strategies of design organisation or management. This study comes up with some descriptive concepts that invite further investigations into generic patterns of teamwork in design.

The remainder of the chapter is organised as follows. The next section gives observations of three historical cases as examples of cooperative architectural modelling. The third section presents an exposition of the aspects of the group dynamics in collaborative design based on the preceding case studies. The exposition leads to two conceptual frameworks for describing cooperative design modelling, which centre around the interrelations between common images and distributed domain design developments. Finally, in the fourth section, the implications of the current work and further analyses to be carried out next are discussed.

2.2 Examples of Teamwork in Design Modelling

Architectural modelling can take place in various dimensions, allowing for a wide variety of collaborative involvement. What follows in this section is an introduction to this variety illustrated by three case studies. The first case shows a convergence of two individual design worlds employing different conceptual schemes in modelling a fountain

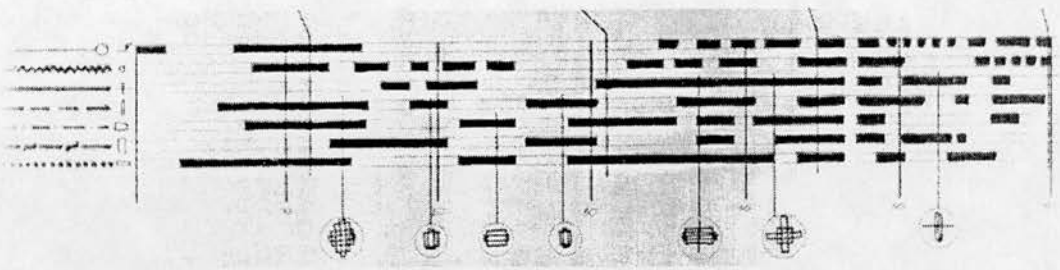


Figure 2.1. The landscape designers' introducing and operating with *score* in modelling fountain patterns and actions over a period of time.

design. The second shows the overlaying of two-dimensional diagrams constructed by at least three engineering disciplines, for re-engineering a large industrial building. The third case illustrates a three-dimensional funicular model that was devised commonly, but used differently by individuals who participated in a church design project.

2.2.1 Case 1: Between score and diagram

Design project: Seattle Center Fountain, Seattle, USA, 1962–1964.
Participating design worlds: (a) waterscape design by two landscape architects (Lawrence Halprin, Curtis Schreier); (b) fountain engineering by a mechanical engineer (Daniel Yanow) [Hal69].

The scoring and diagramming spaces

- The Landscape Architects (LA) used a particular representation scheme called "score" for modelling fountain patterns and actions in a temporal frame (Figure 2.1, drawing taken from [Hal69, p. 56]). A score has two dimensions: one for regulating multiple temporal sequences, represented in certain lengths of bars; the other for configuring spatial structures of different fountain stages (platforms), represented as point, square cross, rectangle etc. By manipulating the bars, a score reveals different compositions of active fountain stages against the inactive ones over a period of time.

- The Mechanical Engineer (ME) used "diagrams" to model mechanical components for piping, jetting, and sprinkling design (Figure 2.2, drawing taken from [Hal69, p. 56]). A pool piping grid was composed in a system of graphical symbols, corresponding to a set of design objects whose attributes were specified in

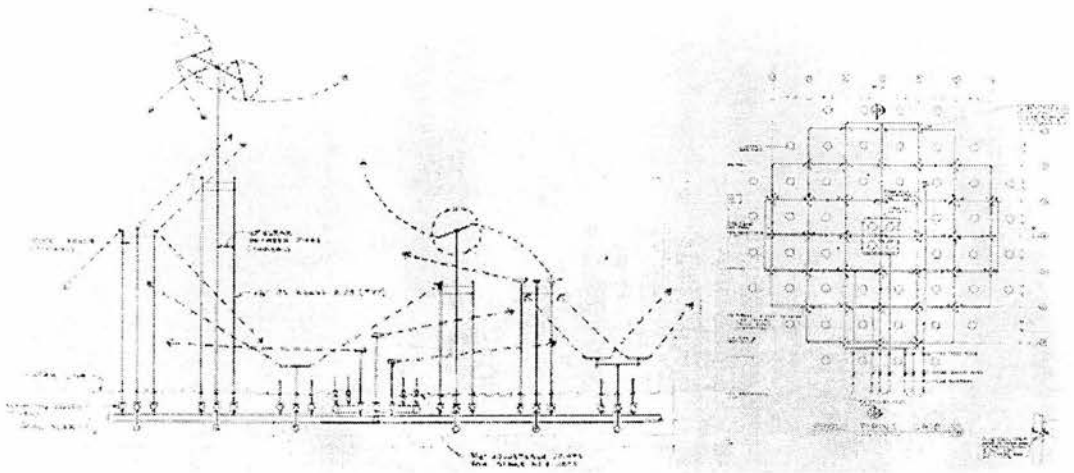


Figure 2.2. The mechanical engineer's introducing and operating with *piping diagrams* in modelling the behaviours of the mechanical components.

words and numerals. In relation to the piping grid, a mechanical section was constructed to convey sectional information. Due to the correspondence set up between the mechanical components and the graphical symbols, the ME could virtually change his idea of the attributes and relations of particular design objects by manipulating parts of the diagrams.

A common space for projecting water effects Figure 2.3 (drawing taken from [Hal69, p. 56]) shows a series of graphical expressions of *squiggles* spreading over a regular *grid*. This evidence implies that a common modelling space shared by LA and ME was being used, combining the designs in the score and in the diagram, by which a sequence of water effects can be *projected*. Here we see an example of a set of *common images* generated, allowing for *interpretations* of the design consequences from various viewpoints. It is clear that the images of water effects can be interpreted both in LA's view—the actions of fountain stages as scored over a time span, and in ME's view—the fountain kinematics concerning the motions in pipes, jet heads, and sprinklers, as configured in the piping grid and mechanical section.

Interrelations between the score and the diagram Given the above evidence, two interrelations between scoring, diagramming, and projecting spaces are worth noting, which yield further explanations of what constitutes participation in developing

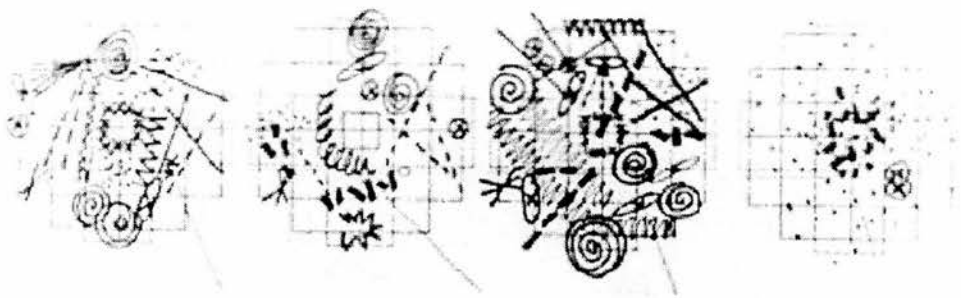


Figure 2.3. The graphical indications of a shared fountain modelling formed by a combination of LA's scoring and ME's diagramming, which can project water effects, allowing for different interpretations.

the fountain design:

- Sequences of water effects at particular moments cannot be projected solely in LA's scoring space nor in ME's diagramming space; the ability of projecting these effects is conditioned by knowing what fountain stages are active and what mechanical devices are operating on those active stages, plus how they will behave—a convergence between two individual modelling spaces whenever a projection is undertaken.

- Modelling actions taken in individual spaces change not only the state of score or diagram but also the state of the common image when projected; ME may take further actions upon what he interprets as changing water effects propagated from LA's actions in changing the score, and vice versa—communication and coordination are called for to resolve disagreements or conflicts thus arising.

2.2.2 Case 2: Cooperation through overlay diagramming

Design project: Cummins Research and Engineering Center, Indiana, USA, 1964–1968.

Participating design worlds: (a) Structural Engineering (SE) (The Engineers Collaborative), (b) Lighting Engineering (LE) (William Lam Associates), (c) Mechanical Engineering (ME) (Cosentini Associates). The main design issue is centred on how to "rearrange ductwork to the structure and to baffle the indirect light sources" [Lam77, pp. 125–129].

Figure 2.4. Multiple diagramming spaces in different layers showing the participants' heterogeneous coding systems for modelling aspects of the building design.

Distributed diagramming spaces Each engineering discipline had its own object-based diagramming space. There were at least three domain-oriented diagramming spaces participating in the project: SE, ME, and LE (Figure 2.4, drawings taken from [Lam77, p. 126]). Each diagramming space employed a special coding system to represent the modelled building components.

Collaborative environmental design through overlaying According to Lam, the group processes evolved a "fishbone layout" which proved to be economical and satisfactory to all participants [Lam77] (see Figure 2.5, drawings taken from [Lam77, p. 126]). Clearly, the emergence of the fishbone image is conditioned by the participants' continuously *overlaying* their individually produced diagrams. But an important point is that the common projection, here in the form of a 'fishbone', provides the members of the design team with a common focus for expressing agreement.

Articulation of common images In the previous case we saw an example of two participating design worlds related to each other by the projection of final design effects. The current case, however, shows that participants can further *articulate* a projected common image into various parts that play different roles or functions in the design (the spinal cord and ribs of a fishbone, in this example). Differentiated portions of a common image are then distributed to serve individuals' further development of

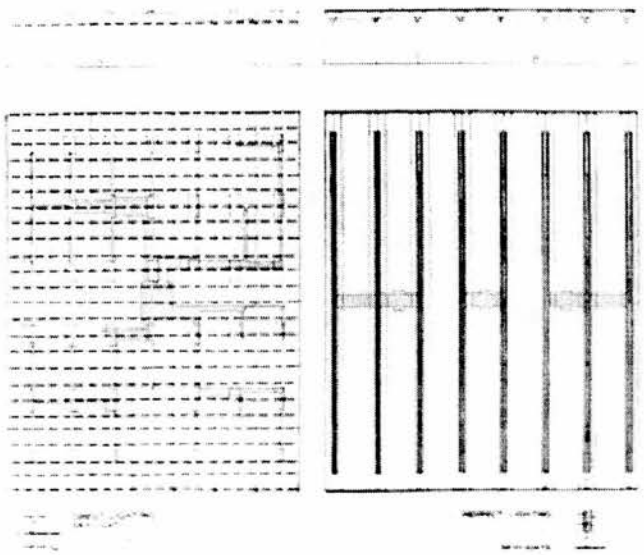


Figure 2.5. The combined images of structural, mechanical, lighting design solutions get evolved through the participants' overlaying domain design developments.

domain design solutions. Given the evidence that common images can be evolutionary (i.e., the designers evolve their ideas about what the 'fishbone' will look like and how it will work), it is important to recognise the process of group articulation. That is, parts of an existing common image get identified and developed by different individuals, which, in turn, may contribute to a new state of the common image. Teamwork in design is therefore maintained by a *to and fro* relation between the individual and the common.

The method of overlay diagramming or drafting has been developed for sometime in design practice. Woods and Powell have documented the method and recommended it as a standard team practice [WP87]. In general, we can point out that the overlay method is used to enable collaborative design practice in the following aspects:

- *overlay diagram construction:* A participant can construct diagrams on top of extracted common images which may contain parts of diagrams drawn by other designers working in different aspects.

- *overlay design checking:* Participatory design can be evaluated by checking overlaid consequences according to certain criteria such as detection of spatial clashes.

- *overlay design amendment:* A participant can modify parts of diagrams by referring to the diagrams underlaid for various purposes (e.g., geometrical, structural, or aesthetical etc.); and one designer's amendments may cause related changes to be made by others.

2.2.3 Case 3: Funicular modelling revisited

Design project: The Colonia Güell Church, Barcelona, Spain, 1889–1914.
Participating design worlds: (a) site planning and structural form by architects (Antonio Gaudì, Jose Canaleta), (b) structural engineering by a civil engineer (Eduardo Goetz), (c) ornamentation by a sculptor (Juan Bertran) [Mar79].

The funicular modelling space An upside-down funicular model was constructed by the design participants at the inception of the project. According to Collins *et al.* [CN83], this large 3-dimensional model, which was shared and manipulated by all participants for different design tasks, had the following distinctive types of model components (Figure 2.6, picture taken from [CN83, Fig. 39]):

- *cords* hung in loops corresponding upside down to the placement and shapes of the piers and arches of the building's vault;
- several pieces of irregularly shaped *boards* fixed onto the structure of the workshop, representing contour lines of the building site;
- *weights* made of pellets contained in small sacks (measured in the scale of $\frac{1}{10,000}$), when attached to the hung cords, distorting the cords' catenary curves into funicular polygons;
- *fabric* (tissue paper) draped onto the web of funicular polygons, representing the volumetric effects of the building exterior;
- a set of *domain-neutral objects* made of jointers, hooks, and clippers, which do not represent any particular components of the building design but function importantly in connecting the model objects and in manipulating parts of the funicular model[1].

From the funicular model to the distributed drawing spaces Apart from the funicular model constructed by the group, there are other special modelling spaces

[1] Jointers were used for attaching weights to cords; hooks for connecting the ends of cords to particular locations on the boards; clippers for clipping cords together at various heights (bifurcation).

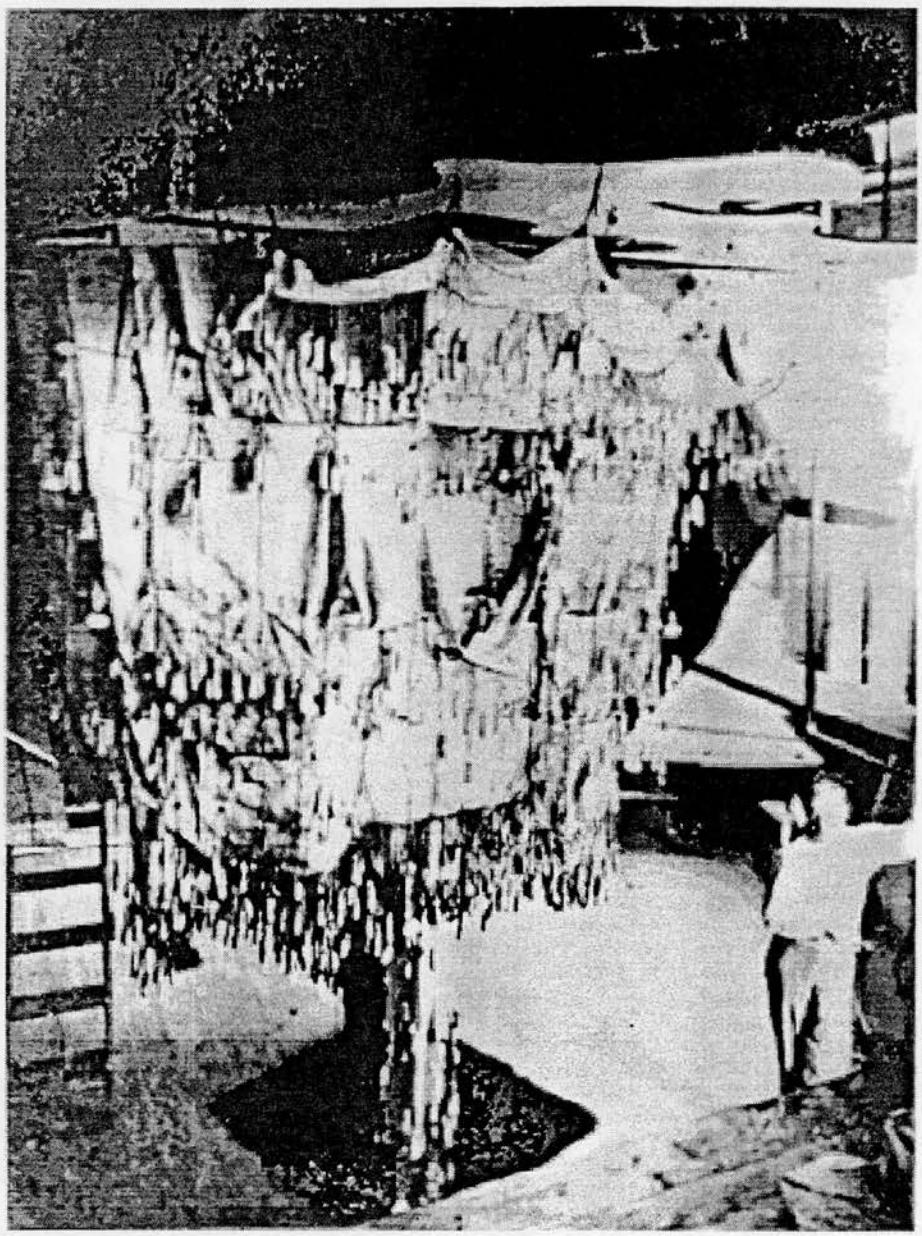


Figure 2.6. The funicular model constructed for the Colonia Güell church project as it hung in the workshed.

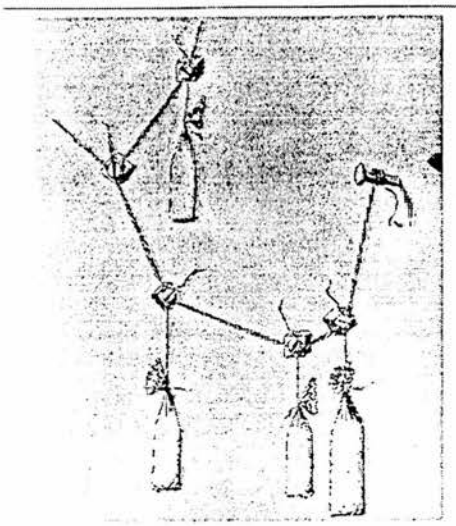
created and used by different individuals. As reported in the research literature, the graphical evidence shows that the distribution of special modelling spaces includes at least the following:

- The civil engineer's structural calculations: the distribution of loads in space and the thrusts of force lines were calculated by the engineer in a 2-dimensional vector space; to him, the funicular model was seen as a 3-dimensional illustration of planar and sectional *graphic static calculations* (Figure 2.7).
- The architects' sketching out of the exterior and interior spaces: photographs of the exterior and interior of the funicular model were taken and turned right-side up by the architects as the underlay information for modelling the locations, proportions, and shapes of opening (the fenestration of the building) (Figure 2.8 and Figure 2.9).
- The sculptor's sketching out the ornamentation: the sculptor was concerned with the design of sculptural objects as the ornaments for the building's exterior and interior; like the architects, he was able to take photographs of the funicular models for his own design purposes and try out overlay sketching (Figure 2.10).

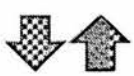
By gathering the different sets of pictorial evidence, Figure 2.11 (drawings and picture taken from [CN83, Fig. 41, Pl. 57, Pl. 55A, and Pl. 59]) shows an overall view of the collaborative setting for the Güell church design: first, there is a common workspace used to construct and change the funicular skeleton; second, there is number of separate workspaces created and used by different participants for domain-specific design developments; and thirdly, the distributed modelling spaces are related to the funicular modelling space in one way or another.

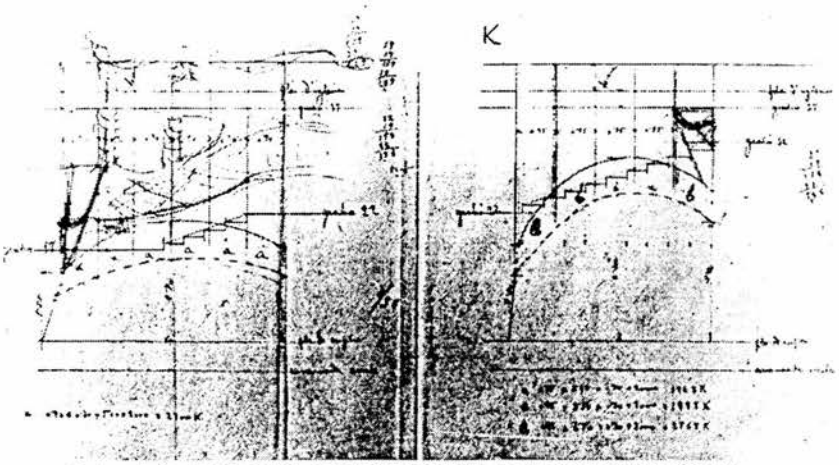
Group interaction in the funicular modelling space Given the above observations, several accounts can be given of what makes the funicular modelling space a shared workspace for the design team, and how the shared model serves as evidence of interaction between the participants:

- First of all, the funicular modelling space was continuously developed and used by the design team for supporting long term participation; it was reported that the participants collaborated on delicate exploratory work lasting over 10 years [Mar79].
- Though all participants shared the same construction of a structural form, the shared funicular modelling space allowed them to manipulate parts of the skeleton



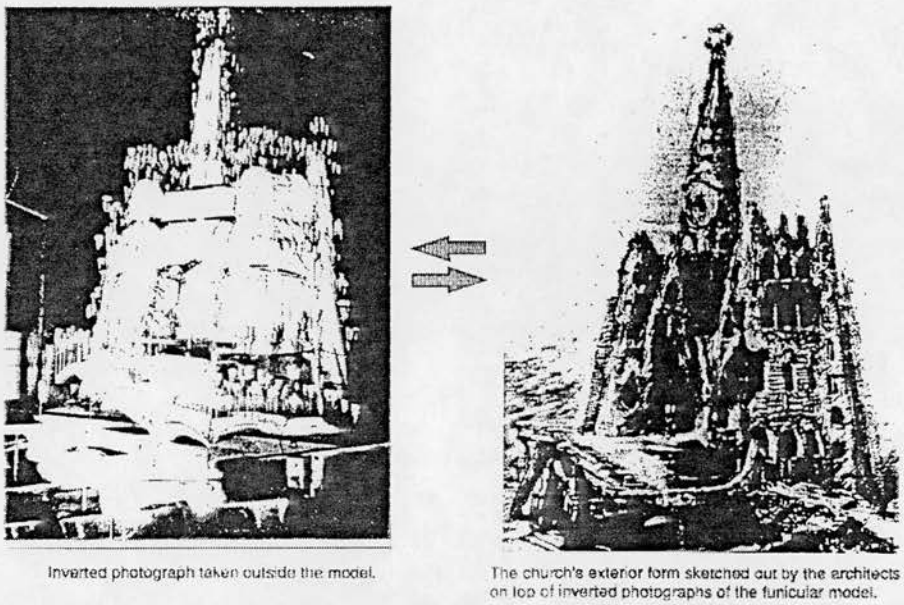
The attachment of weights to cords hung in loops and the resultant distortion of the cord loops into funicular polygons.





A sheet of graphic-static calculations for the ramp-porch of the church produced by the civil engineer.

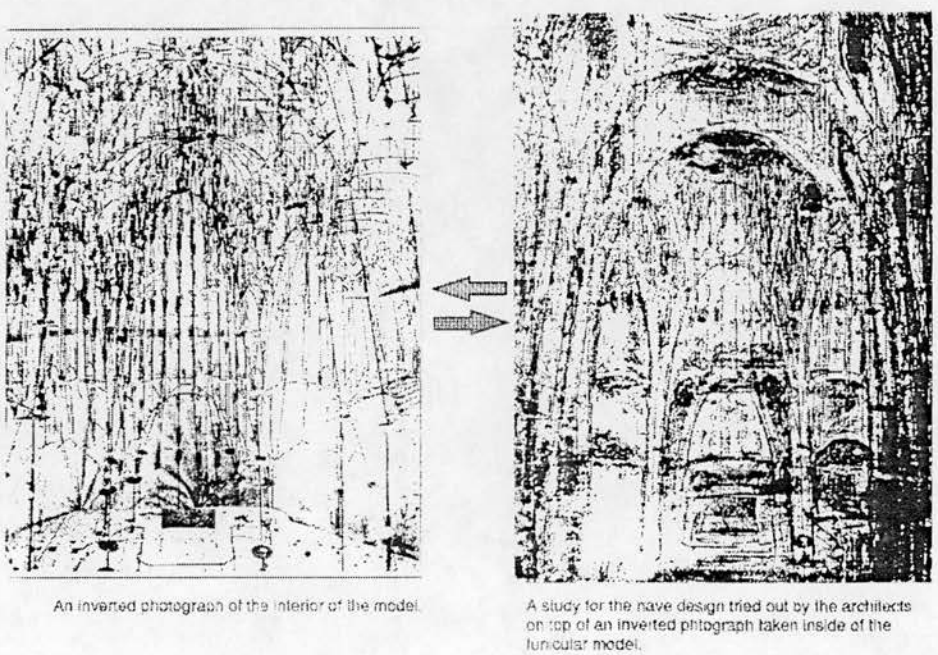
Figure 2.7. The civil engineer's way of working out detailed structural calculations in relation to the funicular structure.



Inverted photograph taken outside the model.

The church's exterior form sketched out by the architects on top of inverted photographs of the funicular model.

Figure 2.8. The architect's way of working out the exterior of the church in relation to the funicular structure.



An inverted photograph of the interior of the model.

A study for the nave design tried out by the architects on top of an inverted photograph taken inside of the funicular model.

Figure 2.9. the architect's way of drawing out the interior of the church in relation to the funicular structure.

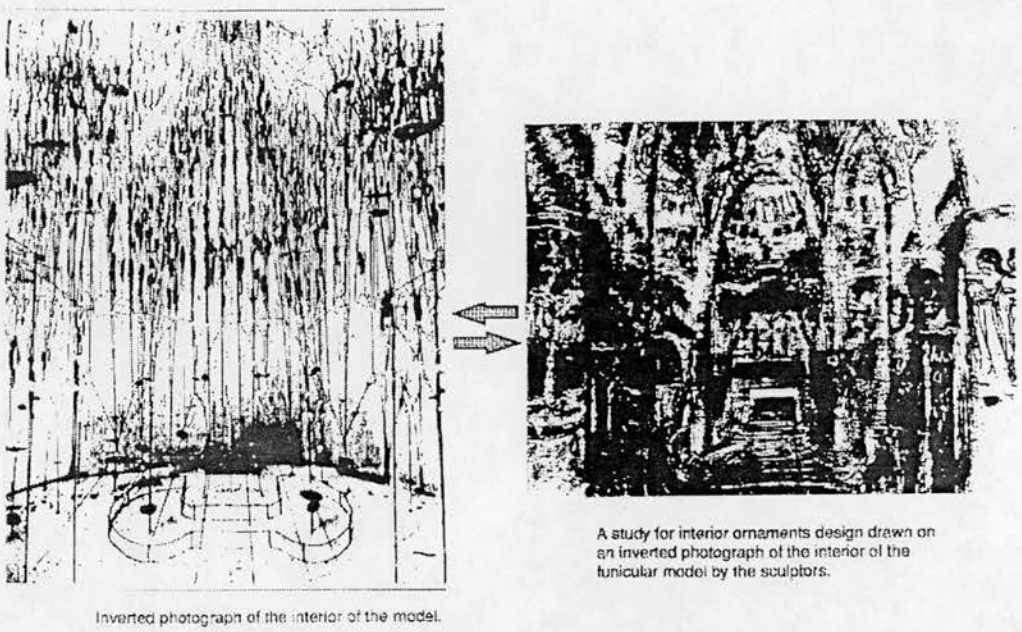


Figure 2.10. The sculptor's way of sketching out the interior ornamentation scheme in relation to the funicular structure.

for reasons other than the strictly structural.[2]

- For any state of the model, the participants could have individual interpretations and derive design information from, perhaps, different measurements; and the information derived further served as the basis for the individuals to elaborate individual design models distributed over several work settings.
- The fact that the earth's gravity was one of the (direct) forces in shaping the model can explain how the group interaction could be coordinated by the shared modelling space. Through the action of G-force, the model constructed and manipulated always conforms to the physical law of *funicular structure* [Sch80]. Therefore, a modelling action taken by an individual, for whatever reason, can stimulate other team members' interpretations and evoke actions in response to the changing state of the funicular model.

[2]For instance, for the purpose of site planning, cords can be shifted to different hooks or by moving the hooks around the board; for modifying fenestration design, cords can be bifurcated at various heights by sliding the clippers along the force lines; for changing structural form, loads can be redistributed in space by controlling the number of pellets in sacks or by displacing the sacks' jointers to different positions on cords.

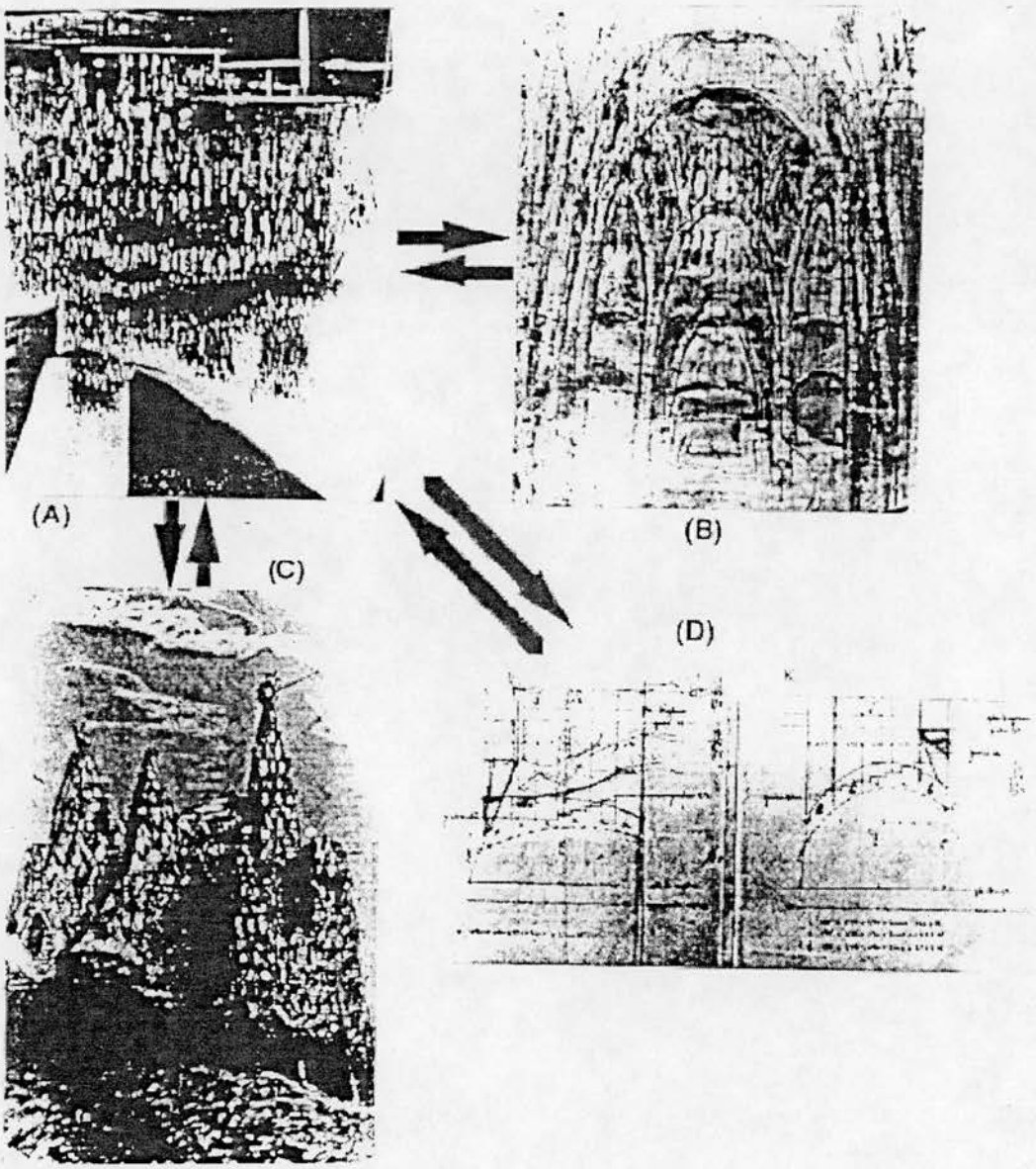


Figure 2.11. An overview shows a number of distributed workspaces that participated in the Güell church design project: (a) the funicular model constructed in the common workshop, (b) an inverted photograph taken inside the funicular structure on which the sculptor's ornamentation design was based, (c) the church's exterior design sketched out by the architects on top of inverted photographs taken outside the model, and (d) force lines constructed by the civil engineer on a projected elevation for structural calculations.

2.3 Group Dynamics in Collaborative Design

Given the various examples of diagrams, drawings, and models observed above, can we extrapolate some forms or processes of communication and coordination that must have taken place for the design teams to arrive at a certain integrity in the design results? In the source materials that we have seen, evidence of how the design developments are related to the interaction among participating design worlds is not explicit. The aim of this section is to explore the hidden dimension of communication embeded in these cases, and to see if we can establish some general notions about "group dynamics" in collaborative design. The general notions arrived at here will serve as an ontological basis for more elaborate analyses of the approaches to teamwork in design in Chapters 4 and 6.

However, it should be made clear from the outset that the group dynamics searched for here has a different emphasis from that of social psychological enquiry, which is concerned more with the knowledge about social psychological forces associated with groups [CA68, p. 4]. In dealing with creative collaborative design, we think it more appropriate to look at the *information* factors associated with teamwork. After all, what we have seen is clearly designers' creating, sharing, distributing, integrating, and changing design information that represents the parts of artefact being designed.

To unravel a general picture of group dynamics in collaborative design, it is important to make the starting point clear. We therefore discuss first what general conditions and goals of teamwork can be abstracted from the case studies. Following the premises of teamwork in design, we introduce the concepts of *modelling spaces* and *modelling acts*. In conjunction with the cases studied earlier, a combination of the modelling spaces and some example modelling acts leads to the notions of *communicative acts*. It is with these abstractions that we attempt to outline some aspects of group dynamics in collaborative design. We will give more detailed expositions of the ontological terms introduced here in later chapters.

2.3.1 Some general conditions and goals observed

Though adopting different modelling approaches to the developments of design solutions, the preceding three examples do suggest some general conditions and goals of collaboration. To develop a general account of communication and coordination among members of a design team, the following initial conditions need to be taken into account:

- Specific or concrete goals of collaboration (e.g., definite descriptions or depictions of final design products) are unknown to or cannot be clearly defined by any

participants at the outset;

- Heterogeneous systems of representation and action employed by individual members are necessarily involved[3];
- No predefined scheduling schemes can be applied to determine how participating design disciplines should coordinate with each other during collaborative sessions, i.e., the strategies for specifying and satisfying precedence constraints of fulfilling cooperative processes cannot be determined by particular individuals in advance.

Working with the above conditions, designers often commit themselves to producing sorts of design information in the forms of sketches, drawings, physical models, and specifications etc. These are the outcomes of individual as well as group working. More specifically, the production of two kinds of information can be generally recognised as the goals of teamwork:

- The information conveys shared conceptions of *unity* in design artefacts to be realised via construction in the real world. Presumably, design expressions of unity can be reached collectively by participants on the basis of, perhaps, their common life experiences and knowledge in dealing with general issues, such as form, movement, human biological or ecological needs, and so on.
- The information contains separate records of *specifications* addressing requirements occurrent in particular design domains. The design specifications are produced on the basis of separated individual specialisms in dealing with technical and detail designs, such as structural, lighting, mechanical services, and so forth.

Based on our case observations, we propose that the two goals are parallel to each other in the sense that no one goal is fully determined by the other. At best, we can say that they are influenced by each other in the course of project development. Now this is an important observation since it requires a more dynamic view of the potential working relations among team members. Metaphorically, we are given two open-ended goals interacting with each other. If some of the dynamisms can be brought out from the general picture, we may be in a better position to further spell out the basic requirements of engaging in creative teamwork in design.

[3]The heterogeneity appearing among participating design worlds is what we observe externally; it should be admitted that there is also (hidden) homogeneous similarity among individual design worlds which makes understanding across heterogeneous differences possible. Not to mention that the co-existence of the two ends is certainly a simplified view of reality.

To start with, we shall return to the basic view of design as modelling complex objects. In theory, the activity of design modelling can be considered to have two components: defining a modelling space, and acting in the defined modelling space.[4]

- design concepts and constructs are continuously introduced and (re)structured as the basic workspaces used by an individual or a group—the aspect of forming *modelling spaces*; and
- shapes and non-shape properties of design artefacts are substantiated, manipulated, and evaluated iteratively by an individual or a group—the aspect of performing *modelling acts*.

2.3.2 Modelling spaces

In a longer term collaborative design process involving heterogeneous modelling spaces, as shown in each of the examples, a clear distinction can be made between the modelling spaces that are formed and used by an individual and spaces that are formed and used by all participants (i.e., the team). Therefore, it can be said that there are multiple Individual Modelling Spaces (IMSs) which are physically and/or functionally separated from a Group Modelling Space (GMS). GMS and IMSs are constructed to hold the creations and modifications of *common images* and *domain design expressions* respectively.

Common images in a GMS

It is repeatedly shown in all the three cases that participating designers' working with heterogeneous design worlds does not prevent their achieving some design expressions that can be commonly shared by all the participants. This is known to us by seeing the sequence of squiggles drawn in the fountain design project, the fishbone image evolved and reported in the engineering research center project, and the state of the funicular structure in the Güell church project. Taking in various forms, expressions of common images are created and used to serve either as generic structures (e.g., the funicular skeleton) or as specific instances (e.g., the fountain squiggles), which allows for individual participants to apply different modelling actions.

[4] 'Modelling Space' introduced here is not to be confused with 'Design Space' used in, mostly, AI approaches to design automation. Given an initial condition, a design space can be generated by a generative system comprised of a set rules (grammar) plus a reasoning or searching strategy. A design space contains a finite number of design solutions that are syntactically sound to the system's grammar.

Constructing common images It can be observed that, embodied mainly as graphical objects, common images can be constructed in a group modelling space in the following two approaches:

1. A common image is constructed jointly by all participants employing a shared representational and operational system; serving as a generic conceptual structure, a common image allows for each participant's deriving and distributing its parts into individual modelling spaces for different modelling purposes.

2. Common images are found by participants' combining and integrating domain-specific design expressions with, perhaps, heterogeneous underlying conceptual structures; serving as outcomes resulting from participation and coordination, common images are inspected and evaluated by individuals to reflect on the design consequences from particular viewpoints.

Changing common images Given an existing common image, its state is subject to unpredicted changes. In respect of how it is formed (in either of the two ways discussed above), changes onto the state of a common image can be effected in two ways:

1. Changes can be made directly to parts of a common image by any participant, if it is formed in a GMS using a common representational and operational system; changes made by one individual to parts of a common image may have related consequences as seen in other participants' IMSs.

2. Changes can be effected indirectly to parts of a common image, if it is formed by combining and integrating parts of heterogeneous design expressions produced by participants. That is, the state of a common image held in a GMS gets updated whenever one or more participants modify parts of their domain expressions. Therefore, changes to a common image caused by one individual in his or her IMS may stimulate further design changes by other participants.

Domain Design Expressions in IMSs

The design of complex things like buildings can seldom reach an executable state without continuous development over months or even years. Each of the case studies clearly shows that a design project can generally not be accomplished within a single profession's judgement and technical specialisation. This means that design developments are often carried out in distributed work processes and settings. In continuing our exposition, it is considered that design developments lead to the generation of design

expressions, and these expressions tend to be domain-specific in the sense that they may contain information only fully comprehensible to the persons from the same design disciplines. Being different from the modelling of common images, domain design expressions are created and stored in a number of logically and/or geographically distributed individual modelling spaces.

Constructing domain design expressions Regarding the construction of domain design expressions, another abstraction from the case evidence can be made as follows:

1. Domain design expressions are developed by elaborating parts of a common image that have been proposed in a GMS. Participants interpret the state of a common image from different perspectives and trace down *derivative images* for individual purposes. The derived images are then imported into IMSs and serve as the basis for further design elaboration, using whatever domain methods that are intended by the individuals.

2. Design expressions addressing particular design domains are firstly developed in participants' IMSs. The initial developments may be independent from one another; but at later stages, some of these expressions may be brought by the individuals into a GMS, serving as the basis for joint construction of common images.

Changing domain design expressions In respect of how they may be constructed in IMSs as discussed above, domain design expressions can be modified in two different ways with the following consequences:

1. In cases where domain design expressions are developed on top of the information derived from a common image, changes in domain design expressions need to be effected and reflected back in parts of the common image. The changes thus made may consequently change parts of other derivative images that are distributed in other IMSs.

2. In cases where a common image is formed on the basis of combining and integrating domain design expressions, changes targeted at the latter can be made directly in the participating IMSs. The changes thus made may lead to an updated state of the common image, which in turn can motivate further changes to be made in other participants' domain design expressions.

Coupling of modelling spaces

In the above, four concepts in terms of group and individual modelling spaces, common images and domain design expressions are formulated. These basic constructs may constitute what we think the artefactual factors of cooperative design modelling. It is for purposes of clarity that they are described as separate conceptual entities. As has been implied, there are intrinsic relations among these factors. Particularly, in considering the construction and subsequent changes to common images and domain design expressions, there exist necessary exchanges between group and individual modelling spaces. For these interrelations, the fifth concept —coupling of modelling spaces—is now formulated more concisely:

- $(\frac{\mathcal{CI}}{GMS} \rightarrow \frac{\mathcal{DDE}s}{IMSs})$. That is, the construction and change of Common Images ($\mathcal{CI}$) in a group modelling space leads to (or constrains) the developments of Domain Design Expressions ($\mathcal{DDE}s$) in distributed individual modelling spaces. The funicular modelling and the graphical modelling in other aspects shown in Case 3 is an example of such a coupling between group and individual modelling spaces.
- $(\frac{\mathcal{DDE}s}{IMSs} \rightarrow \frac{\mathcal{CI}}{GMS})$ That is, the developments of domain design expressions in distributed individual modelling spaces lead to (or constrain) the construction and change of common images in a group modelling space. The coupling of the scoring, diagramming and projecting spaces shown in Case 1, and that of the three diagramming spaces together with shared overlapping space shown in Case 2 are two examples of this type of modelling spaces coupling.

2.3.3 Modelling acts

Following on the artefactual aspect of design modelling just described, we explore what *acts* designers perform in the course of modelling design objects. It is widely acknowledged that there are no 'correct' architectural designs. Given the same design task and tools, it is probable that very different outcomes will come from different designers. A reasonable account of the differences may be formed in the element of action. It is through an individual's actions on artefacts that design objects show their essential character. Therefore, as another conceptual deliberation, the execution of modelling actions constitutes what we call *modelling acts*. Not intended as an exhaustive list, some of the examplar acts seen from the design as modelling point of view are:

- *Representing* — involving, first, a collection of *primitive objects* (constructs) which represents analogically or symbolically the corresponding elements of a design

artefact in the real world; secondly, specifying how instances of the primitives are related in terms of what *operations* are applicable to the constructs. The act of *representing* leads to (explicit) *conceptual structures* in a modelling space.

- *Mapping* — the act of translation and integration of (parts of) an existing conceptual structure to (parts of) another conceptual structure.
- *Constructing* — (given a formalised conceptual structure) the act of applying operations onto selected primitives for creation and modification of $\mathcal{CI}$ or $\mathcal{DDE}s$.
- *Querying* — (given a formalised conceptual structure) the act of applying operations onto parts of $\mathcal{DDE}s$ or $\mathcal{CI}$ for evaluating design instances.

2.3.4 Communicative acts in collaborative design

As a final part of our exposition, a matrix of *communicative acts* in collaborative design is proposed (See Figure 2.12). The matrix is constructed by putting together the elements drawn from the preceding descriptions of the artefact and the action aspects of collaborative modelling. A key point is that, when situated in the settings of coupled group-individual modelling spaces, modelling acts become design actions that require communications among designers working in different domains (See Figure 2.12, note that the circled numerals shown in the matrix correspond to the enumerated items where more detailed explanations are given in this subsection.)

The matrix formed above actually generates eight entries of communication in collaborative design. Obviously, the number of entries is not significant, since more communicative acts can emerge if further modellings acts are added. But a more immediate task is to focus on what the current entries tell us about group interaction. For this, a closer examinination of the matrix items is given below.

1. (Representing in $\frac{\mathcal{CI}}{GMS} \rightarrow \frac{\mathcal{DDE}s}{IMSs}$) The modelling act of representing becomes communicative among participants in the setting up of *shared generic structures* in a group modelling space. This involves the team members' sharing of *group primitives* and the operations for manipulating instances of the primitives such that common images can be evolved.
2. (Representing in $\frac{\mathcal{DDE}s}{IMSs} \rightarrow \frac{\mathcal{CI}}{GMS}$) Conceptual structures are set up by participants in individual modelling spaces; group communications are required when parts of individual conceptual structures are combined and integrated into common sets of constructs in a group modelling space.

Spaces Coupling / Modelling Acts	$\frac{CI}{GMS} \rightarrow \frac{DDEs}{IMSs}$	$\frac{DDEs}{IMSs} \rightarrow \frac{CI}{GMS}$
Representing	①	②
Mapping	③	④
Constructing	⑤	⑥
Querying	⑦	⑧

Figure 2.12. When situated in the two settings of coupled group-individual modelling spaces, modelling acts become communicative acts in collaborative design.

3. (Mapping in $\frac{\mathcal{CI}}{GMS} \rightarrow \frac{\mathcal{DDE}s}{IMSs}$) Participants apply deductive or projective means on states of common images and produce derivative images for individual uses. Group communications come into play when a member's mapping parts of a common image *overlaps* (spatially or logically) with another member's mapping.

4. (Mapping in $\frac{\mathcal{DDE}s}{IMSs} \rightarrow \frac{\mathcal{CI}}{GMS}$) Participants translate parts of individual conceptual structures set up in one design domain into another. An individual's knowing what parts to be translated and translate to what relies on communication for resolving interpersonal/group purposes.

5. (Constructing in $\frac{\mathcal{CI}}{GMS} \rightarrow \frac{\mathcal{DDE}s}{IMSs}$) Participants apply operations onto group primitives yielding states of common images in a group modelling space. Owing to the spaces coupling, events of creating/changing parts of a common image has consequences that may affect members' creating/changing domain design expressions distributed over a number of individual modelling spaces. Group communications are called for in case some of the participants refuse to accept the changing states of design expressions that are their domains of concerns.

6. (Constructing in $\frac{\mathcal{DDE}s}{IMSs} \rightarrow \frac{\mathcal{CI}}{GMS}$) An individual applies operations onto his or her own conceptual primitives yielding instances of domain design expressions. Owing to the spaces coupling, the events of creating/changing domain design expressions

in individual modelling spaces may lead to changing states of a common image that are observable to all participants. Group communications are called for to resolve disagreements or conflicts thus manifested in the changing states of common images.

7. (Querying in $\frac{\mathcal{CI}}{GMS} \rightarrow \frac{\mathcal{DDE}s}{IMSs}$) In general, querying is to ask for information about the consequences of making or changing states of common images or domain expressions. In this case, designers are concerned with the consequences of changing parts of common images modelled in a group space. The acts of querying become communicative since a true picture of modelling consequences can only be delivered by gathering the states of domain design expressions affected by the events.

8. (Querying in $\frac{\mathcal{DDE}s}{IMSs} \rightarrow \frac{\mathcal{CI}}{GMS}$) In this case, designers are concerned with the consequences of making changes in domain design expressions. Group communications are called for due to a presentation of the current state of common images, which in fact requires participants' providing the latest states of domain design expressions constructed in their individual modelling spaces.

2.4 A Spectrum of Possibilities

Given the above exposition, we may now give a list of factors that can be associated with group interaction in collaborative design:

- *the sharing of a group modelling space* — participants need to share (to know and to operate with) a collection of design primitives and operations to construct shared generic structures, or a scheme of integrated conceptual structures to project shared design metaphors.

- *the construction of common images* — in the case of constructing a shared generic structure, participants need to be constantly aware of the current state of the structure in relation to their local design developments; in the case of projecting shared design metaphor, participants need to provide source expressions jointly so that common images can be projected in their group modelling space.

- *the interdependence between $\mathcal{CI}$ and $\mathcal{DDE}s$* — due to the coupling of group and individual modelling spaces and participants' making changes in the spaces, there are consequent flows of design information between common images and domain design expressions.

- *the communication of design judgements* — designers need to say to each other what he or she thinks of the effects of design changes proposed by others from his or her own domain point of view.

By putting these information factors together, two distinct teamwork patterns of collaborative design seem to emerge from our current discussion. To better present this distinction, we propose to characterise the two teamwork patterns as *structuralist* and *metaphorist*. A diagram summarising the operations of the two teamwork patterns is provided in Figure 2.13.

Seen from the structuralist stand-point, to collaborate on modelling complex objects, common images as shared generic structures built upon group primitives and operations play a significant role in coordinating participants' modelling activities. Group modelling spaces in this case shall function mainly as a shared construction system, making use of some sort or sorts of physical forces. The funicular modelling space is such an example. However, in serving a similar purpose, there can be other form-finding systems which introduce physical or formal laws other than the gravitational one.[5] By mapping parts of common images into derivative images from different design perspectives, domain design expressions can be further developed (or elaborated) in distributed individual modelling spaces. Designers then receive consequences of making changes by querying states of common images which may activate interpersonal coordination.

On the other hand, seen from the metaphorist stand-point, cooperative design is approached by participants' introducing domain primitives and operations with which domain expressions can be modelled in individual spaces. By presenting domain proposals in a public forum, the collaboration for achieving integrated conceptual structures or schemata is initiated. In this case, common images are collaboratively constructed by combining and integrating domain design expressions, using the shared design constructs or primitives and operations. Common images are said to serve the participants as shared metaphors whose states in turn play a role in coordinating design activities across various modelling disciplines. The emergence of squiggles and fishbone are two examples of common images that reveal the consequences of integrating participating domain expressions in certain ways as intended by the collaborators.

Limitations and work to follow

In the above, we have looked for evidence of teamwork in design exclusively from

[5] As far as the construction of spatial structures or skeletons is concerned, there are other eminent formal form-defining systems such as the ancient Greek *taxis* schemas of subdividing a building composition discussed in [TL87], and the more modern spatial grammars devised by Durand [Dur75].

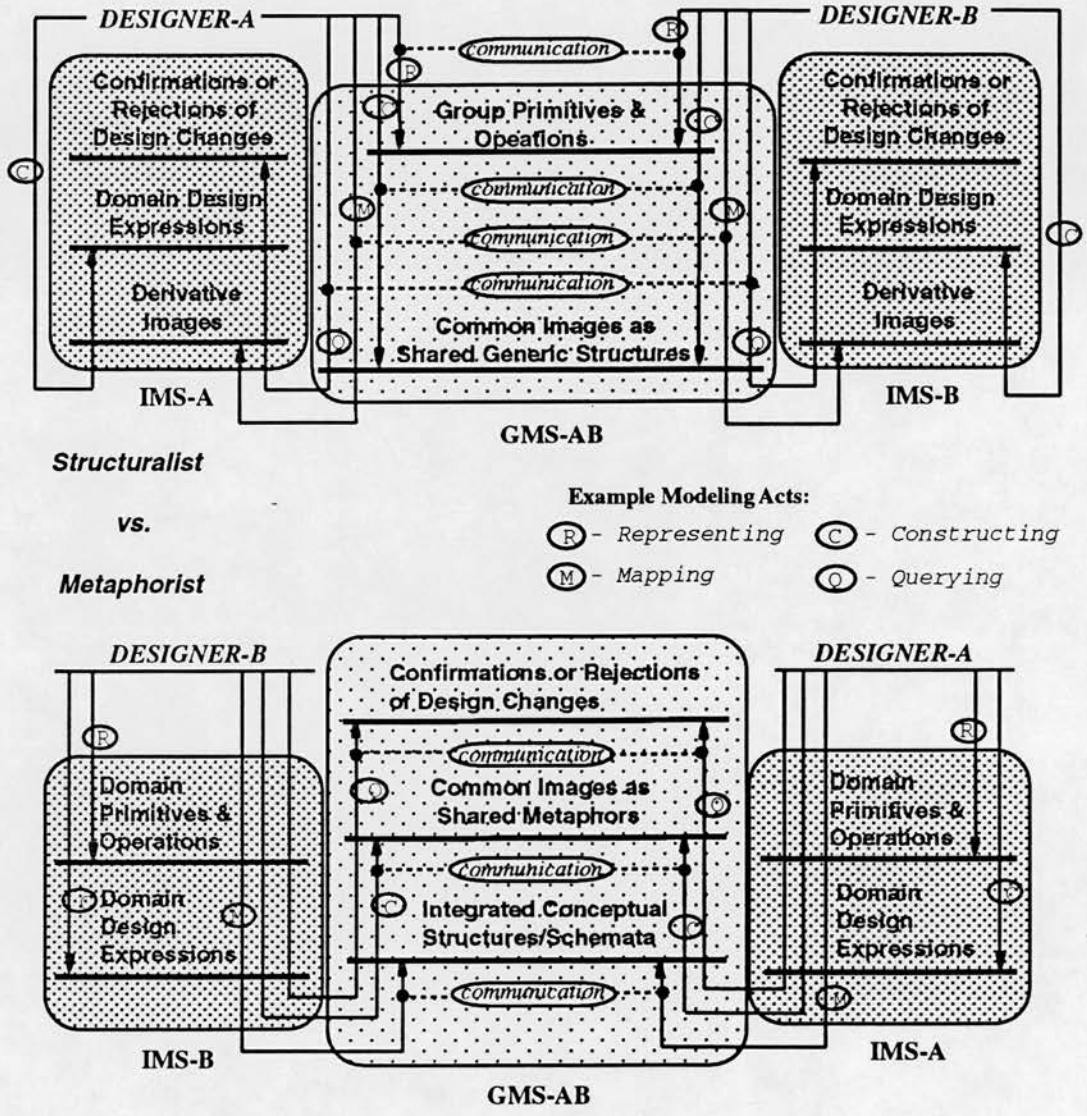


Figure 2.13. Two abstract communication patterns found in the current study of cooperative architectural modelling are characterised as *structuralist* and *metaphorist*. The number of designers indicated is arbitrary. The scaling of 2 to n designers in a design team can be envisaged by viewing this diagram as a 'section of a cylindrical structure'. Seen from this picture, in coordinating modelling activities with other members, an individual's workspace is a combination of his or her own IMS and a GMS.

the design expressions generated among (professional) designers. Collaborative design to many people may include a wider scope of participation including, for example, clients, end users, or governmental representatives, etc. How and what to support teamwork involving both professional and non-professional designers is an important and challenging subject, and it will certainly demand much more efforts to describe the representation and communication requirements in a clear and organised way. Given the limited research resources of the thesis, a research into a wider scope of participation is not considered.

Having admitted the overall limit of this study, our current observation of the dynamism of group interaction in collaborative design has other limitations. Obviously, the number of case studies is limited. The subsequent explanations given and the general concepts derived are, therefore, restricted to the scope where supporting evidence from the case studies can be found. In consequence, the current inclusion of the information factors associated with group interaction may need to be extended when the base of case studies is enlarged. Also, our characterisation of the structuralist and metaphorist patterns can only be considered as, not two alternatives, but two among, possibly, many others that are beyond what we have observed here.

However, the limitations shown here seem not to pose fundamental questions to the validity of the current exposition. What is needed is a more fine-grained structural study into each of the teamwork patterns characterised here. Especially, we need to know more about the aspects of information flow, which will help to deliver a sharper understanding of the representation and communication requirements for supporting inter-personal and domain-crossing interaction in collaborative design. A deeper appreciation of the teamwork structures is also expected to produce useful guidance in exploring the potential computing architectures that support collaborative design. But before delving into the structuralist and metaphorist patterns, we shall go through, in the next chapter, a survey of recent experimentation with communicating and computing tools in supporting group drawing activity.

Summary

At the beginning of this chapter, the aim and nature of our searching for the evidence of teamwork in design is outlined, which appears somewhat different from the existing research done in the fields of design studies and computer-supported cooperative work. From this initial position, the graphical evidence collected from three historical cases are examined. The evidence commonly points to a problematic situation of collaborative design. First of all, participants of a design group often work with heterogeneous

systems of representation and action that correspond to individual schools of design expertise. Moreover, participants have the common goal of achieving overall design unity in parallel to individual goals of achieving technically sound domain design solutions. Given a situation such as this, an interesting issue to account for is how do designers manage to achieve both common and individual goals and what is involved in their doing so.

By extending our basic view of design as modelling, we come to the notions of modelling spaces and modelling acts. Given the case evidence, two categories of design representation are differentiated: common images and domain design expressions. By putting these concepts together, we arrive at a matrix of communicative acts in collaborative design. Eight communicative acts are discussed to show where communications and coordination among group members are needed. In consequence, four information factors associated with group interaction in collaborative design are identified. By drawing the factors and concepts into a framework-like structure, our current case studies end up with a distinction between two teamwork patterns: structuralist and metaphorist, which suggest two general supporting architectures for more detailed studies to follow.

Chapter 3

Experiments in Supporting Collaborative Drawing

Along with recent experiments in the design of communication or computer tools for supporting various kinds of group working, the development of *collaborative drawing systems* has emerged as a notable research area within the field of Computer-Supported Cooperative Work. This chapter reports on a survey of the experiments in collaborative drawing support tools with the objective of reviewing how the issues of supporting *collaborative design* have been addressed by the research prototypes. The survey is presented in three parts: (1) findings from the observations of group interaction in drawing and design activities, (2) a framework for classifying the design issues experimented with by prototype developers, and (3) a categorisation of the current prototype systems by interrelating the patterns of group use observed with the system features classified. The survey indicates that there are currently at least three different strategies for developing collaborative drawing support tools: as media of real-time graphical conversations, as tools used by participants to manage design ideas, and as media of performing in team-room. The differences reflect the existence of diversified understanding and technological responses to what and how human collaboration in design may be supported.

3.1 Background and Objective of Survey

Design as a human activity is pervasive; and designers often work jointly to develop usable and meaningful artefacts. The problem of how to develop communication and computer systems that can support collaborative design or problem solving has become

an active research area, attracting researchers working on various perspectives. In a recent bibliographical survey of the research literature in the field of Computer-Supported Cooperative Work (CSCW), Greenberg [Gre91] has formally introduced the key word *shared workspace*. It is noticeable that a large portion of research work on shared workspaces has to do with building prototypes of *collaborative drawing support tools*. Being perhaps inspired and guided by earlier observational studies of working group graphics and shared drawing space activities, (see [Lak83, Bly88, TL88] among others), researchers have been attempting to design and implement prototypes of shared drawing systems. The requirements for these systems to meet are different from the traditional ones; one of the major goals of implementing shared drawing tools is to facilitate communication and coordination among participants in the course of creating and using technical or non-technical drawings.

However, group interaction in design has been observed, described, and analysed differently due to the various perspectives adopted. There accordingly appears a diversity of understanding as well as assumptions as to what might constitute a shared workspace which enables human communication and coordination in carrying out design tasks. Differences in the basic investigation of the question

> *how are drawings as shared artefacts, and group drawing as shared drawing space activity, related to collaborative design processes?*

have resulted in varied technical approaches to answering

> *what is to be facilitated by shared drawing support tools?*

Though a number of group drawing support tools have been previously reviewed (see, for instance, [Lu92]), this survey, by reviewing a wider range of group drawing tools, is intended to be more comprehensive. Instead of giving descriptions specific to particular system implementations, the objective of our survey is threefold: (1) to identify the important aspects of understanding collaborative drawing and design activities, regarding the original studies that have been made, (2) to present a framework for classifying the design issues being experimented with by the current prototype developers, and (3) to catalogue the features and components of the prototype systems in the survey.

The remainder of the chapter is organised as follows. A discussion of different understandings of group drawing activities from various perspectives is given in the next section. Given the conceptual aspects discussed, Section 3.2 is devoted to a classification of the design issues emerging from the current experimentation in collaborative

drawing systems. In Section 3.3, to give a clear overview of the current status of prototype development, a categorisation which mixes the dimension of the modes of system use with the dimension of system components is presented. Finally, some potential topics for further investigation are discussed in Section 3.5.

3.2 Aspects of Group Drawing and Design Activities

Most developments of collaborative drawing tools were motivated and guided by the current understandings of group drawing and design activities. It is therefore an appropriate starting point to look at what has been said or characterised about these activities. Table 3.1 is a summarisation of nine such studies. The nine groups' studies are chosen because they present original research perspectives and respond to the problems with various prototype solutions. The differences arising here are significant in showing that it is quite possible to have very different angles when characterising what is involved in group drawing activities.[1]

To better understand the various issues raised by these original studies and experimentation, a more detailed comparative study is presented below, focusing on the aspects of *events, information, tools,* and *ownership.* These four aspects are chosen because they are the common conceptual issues that were addressed by the research groups to various extents. Other issues concerning more of the aspects of system design and implementation are left to the next section.

3.2.1 Events: collocated vs. remote; synchronous vs. asynchronous

Since any collaborative drawing or design activity must take place in space and time, the patterns of events can be generally differentiated in terms of four basic spatio-temporal structures[2] (Figure 3.1).

1. *collocated synchronous.* All participants taking part in a shared design project work in the same setting via face-to-face simultaneous interactions. A typical

[1]It may be controversial to include the observations and systems made by Fischer's group in our survey. However, the inclusion is intended to establish a wider concept of 'shared drawing' activities. In particular, as mentioned in the beginning of the chapter, we feel the need for an examination of any possible links between supporting shared drawing activities and supporting collaborative design. To us, Fischer's systems present a distinct attempt to support a form of shared drawing activities, involving participants' indirect (i.e., asynchronous) communication for exchanging technical knowledge. It is evident that, in Fischer's systems, 'graphical construction' is an integral part of the overall collaborative design processes, though it may appear highly task-oriented.

[2]Similar time space matrices have been proposed by Johansen [Joh88] and Ellis and others [EGR91] in their discussions of groupware design

Research Groups	Fields of Observations	Group Size	Research Methodology	Important Findings	Research Prototypes
Lakin et al. (Lakin 1983; Lakin 1988; Lakin 1990)	Informal and formal working group graphics in engineering, geological survey, and computing	One or more facilitators with non-specified group size	A linguistic analysis of working groups' text-graphics images and manipulation	Spatial and temporal structures in manipulation of text-graphics	*vmacs, and Visual Languages for Cooperation*
Stefik et al. (Stefik et al. 1987a; Tatar et al. 1991)	Small teams of computer scientists engaged in face-to-face meetings	2—6 persons	The setting up of an experimental meeting room by networking PCs and a large screen	The effects of turn taking systems on collaborative work taken place in a meeting room	*CoLab Boardnoter Cognoter Argnoter*
Bly et al. (Bly 1988; Minneman & Bly 1991)	Informal drawing in computer user-interface design problems	2—3 designers	Video-audio protocol and a drawing events/ actions/clusters analysis	The uses of drawing surfaces in different collaborative settings	*Commune*
Tang et al. (Tang & Leifer 1988; Tang 1991)	Informal drawing in computer user-interface design problems	3—4 designers	Video-audio protocol and an action-function framework for protocol analysis	The importance of the process of creating and using drawings	*VideoDraw VideoWhite-Board*
Ishii et al. (Ishii 1990; Ishii & Arita 1991; Ishii et al. 1992)	Informal drawing and computer generated images in computer systems design	2—3 designers	Intuitive understanding and building various versions of experimental prototype systems	The importance of integrating social protocols with shared workspaces	*TeamWork-Station ClearFace ClearBoard-1 ClearBoard-2*
Fischer et al. (Fischer et al. 1991; Fischer et al. 1992; Reeves et al. 1992)	Formal coding systems and the structures of expert knowledge in the designs of kitchens and computer networks	not specified	Observing design experts in work, then developing a conceptual framework and a demonstration system	The integration of the design of the artefact (made in combined graphics and texts) and the communication among designers	*JANUS NETWORK-HYDRA XNETWORK*
Lu et al. (Lu & Mantei 1991; Lu 1992)	Multi-layer informal drawing in architectural layout design	2—4 designers	Generating requirements by mapping design behaviours onto skeleton design	Fifteen user requirements for a drawing system to support idea management	*CaveDraw*
Wolf et al. (Wolf & Rhyne 1991; Wolf & Rhyne 1992)	Informal small group meetings for the conceptual design of a "Clutter Collector Robot"	2—10 persons	Developing a pen-based prototype system, putting it in group uses, and getting detailed user feedback	The potential meeting process gains/losses, and the areas for aiding search and retrieval of info.	*We-Met*
Brinck et al. (Brinck & Gomez 1992; Brinck 1993)	The remnants of office whiteboard conversations made for technical versus administrative uses	2—3 designers	Recording and analysing sketches, writing, and retrospective descriptions of conversations	A classification of whiteboard objects and their semantic properties in conversational use	*Conversation-Board*

Table 3.1. A tabulation of nine studies of group drawing and design activities which have direct influences on their own or other prototype system developments.

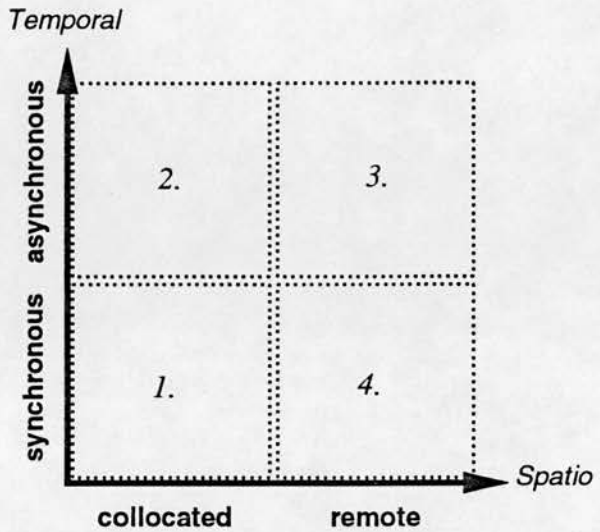


Figure 3.1. A spatio-temporal frame for classifying the events of group drawing or design activity into four basic patterns of collaboration. (The numerals correspond to the enumerated items described in the main text.)

group process of this pattern is a design meeting or brain-storming session which demands close physical proximity for intensive communication among group members. As studied by Lakin ([Lak83, Lak88]), *working group graphics*, operated by one or more operator (or facilitator), plays an important role in aiding collocated simultaneous interactions.

2. *collocated asynchronous.* Given the same projects to embark on, members of a design group are located in the same setting (say, a large design studio) through *indirect* communications over a period of time. Though not being a substantial observational study, a scenario of what might be involved in a day of office work where five staff members take part in a telescope engineering project was described by Lakin [Lak90]. What Lakin considered is that participants work in the same setting but are left alone to concentrate on different aspects of the project. [3] In this way, teamwork is carried out mostly via indirect communication

[3]To this point, 'collocation' seems not an obvious factor for 'collocated asynchronous' being fundamentally different from 'remote asynchronous' if the actual physical distance is taken into account. But, basically, 'collocated' implies that members can see and talk to each other without moving from one place to another. To better illustrate this event type, we may think of a group of people engaged in 'team room' activities, and another good example of collocated asynchronous is 'shift work' (i.e., when one shift passes information to the next shift).

among collocated participants (e.g., passing working documents or CAD files to one another not necessarily involving face-to-face meetings).

3. *remote asynchronous.* In carrying out shared design projects, participants work in geographically distributed settings through indirect communications. As a rationale of designing computer tools supporting design teamwork of this pattern, Fischer and others explained that remote asynchronous collaboration can be commonly seen in modern technologically oriented design projects [FGL+92]:

   "Meetings and other types of direct communication are the commonly used means for coordination and collaboration, but in many situations — especially ones involving long-term collaboration — these are not feasible. Modern design projects can extend over many years and can involve a high turnover in personnel. People who are not in the project group at the same time need to coordinate and collaborate in the design of a system."

4. *remote synchronous.* Geographically, team members are separated (from a few feet to, perhaps, thousands of miles away); but they are enabled to have direct communication while making drawings. Events of this type attract most CSCW researchers' attention. In fact, a large proportion of the research prototypes are dedicated to supporting group drawing activities of this type. Under the spatio-temporal circumstances, collaborative drawing activities are made possible by providing participants with shared *virtual drawing spaces* which emulate as much as can be achieved in a direct face-to-face interaction. In several published studies of supporting remote synchronous group interaction in designing or drawing, the following concepts have been put forward:

   - Drawing activities once shared (simultaneously) can pull designers together and increase their attention and involvement in the design task [Bly88];
   - The processes of creating and using drawings can convey information that is as important as, or, actually, not found in, the resulting drawings [Bly88, Tan91];
   - Designers are themselves skilled at coordinating communication, and the social protocols acquired via the face-to-face communication can serve the needs of constructive collaboration in the shared workspace [Tan89, IO90, IM91, IK92].

3.2.2 Information: action-oriented vs. representation-oriented

Collaborative work in design or problem solving in one way or another has to do with the generation and exchange of information. However, different perspectives have varied considerations of what information is to be captured and transmitted. On most occasions of synchronous collaboration, information useful to group interaction is considered to be *action-oriented*; that is, actions recorded or tracked by devices such as video imaging, shadow projecting, mouse movements and so on, are the kind of information that participants may generate, recall, interpret, and share.

Actions that have been studied in group drawing activities include sketching, writing (listing alpha-numeric text), talking, gesturing, gazing (making eye-contact). The sharing of action-oriented information is claimed to foster and maintain *group awareness*; but how is collective awareness related to the performance of collaborative work? Recently, Ishii and Kobayashi commented on how 'gaze awareness', as supported by the *ClearBoard* system (see Figure 3.4), affects two participants' solving the 'missionaries and cannibals' puzzle [IK92]:

> "Through this experiment we confirmed that it is easy for the players to say which side of the river the partner is gazing at and this information was quite useful in advising each other."

In less direct collaboration, on the other hand, synchronous or asynchronous sharing of design ideas or knowledge conveyed by participants in some representation form is considered more important. When group work involves more technical matters (e.g., the production of technical drawings or the performing of graphical modelling in engineering design), collaboration may necessarily involve some formal system for constructing and interpreting individual or collective expressions. The views in favour of representation-oriented collaboration have presented the following points, arguing how formal representations of information (graphical as well as non-graphical) may serve group interaction in design:

- In Lakin's view [Lak86, Lak90], spatial and temporal structures (schemata) can be observed when designers create and manipulate text-graphic expressions. These structures, on the one hand, limit the kind of expressions and arrangements one may make, while, on the other hand, provide the basis for well-defined spatio-temporal regularities available for automated machine interpretation. Various *visual languages* for collaborative working such as co-authoring, brain-storming, and task structuring can therefore be formally defined.

- In the view of Fischer and others [FGL+92], long-term indirect collaborative design takes the model that coordination of individual work in groups is achieved by the individuals' interactions with 'group memory', which can be represented formally as

  "a collection of shared information repositories containing a cumulative record of rationale, solution components, information about prior projects, and other information resources for collaboration."

Given the group memory represented both in (hyper-) textual and graphical forms, indirect communication among designers can be supported by the computer-based methods of 'argumentation'[4] and 'critiquing'[5].

3.2.3 Tools: homogeneous vs. heterogeneous

The third aspect of understanding collaborative drawing activities is concerned with the use of tools. The problem lies in whether all participants work with the same set of tools, or each of them may need to operate different sets of tools. There appears again a dichotomy between *homogeneous* and *heterogeneous* sets of tools used by participants. Heterogeneity of tools may be due to, for instance, being manual or computer-based, the structures of drawings produced, ways of storing and retrieving data, actions involved in manipulating expressions, or domains of interpretation and so on. Referring to Tang's, Bly's, and Fischer's studies (see Table 3.1), we may first point out two main arguments why the provision and use of homogeneous sets of tools are considered as being sufficient for supporting collaborative drawing and design:

- Synchronous group interaction does not involve domain-specific information or knowledge; i.e., participants converse with each other on design issues readily supported by sufficient common sense such that 'pencil and paper' type tools can satisfy the communication needs of the group.

[4]The argumentation here refers to the argumentation in the Issue Based Information Systems (IBIS) method originally developed by Kunz and Rittel [KR70], and extended by Conklin and Begeman in the *gIBIS* tool [CB88]. According to Fischer *et. al.*, the issue-based argumentation method is intended as an interpretation of the 'reflection' in the design theory of *reflection-in-action* proposed by Schön [Sch85].

[5]As a method parallel to argumentation, critiquing (the generation and sending of *critic messages*) is developed by the Fischer group to identify and explain why a given design construction is inconsistent with the state of group memory (the so called breakdown situations). The critiquing method has been experimented with by the same research group in implementing several cooperative problem solving systems (see [FLMI91], for more details). For the theory and practice of expert critiquing systems, a comprehensive survey can be found in [Sil92].

- Participants come from more or less the same professional background and work within the same design domain; i.e., there is no need for streams of expert knowledge across different design disciplines in the course of collaboration. It is therefore sufficient for all participants to operate the same set of tools, even if it is a highly sophisticated one.

Taking a rather different view, Ishii and Miyake introduce the concept of 'open shared workspace', expressing that "group members should be able to use a variety of heterogeneous sets of tools (computer-based and manual tools) in the shared workspace *simultaneously*" [IM91]. And according to Lakin's observation [Lak90], in ordinary office work, there exist *group-individual mode switching*, (i.e., participants sometimes work as individuals and sometimes as members of a group), and *technical task switching* (i.e., work change between various tasks requiring special technical support). The design of various visual languages for cooperation is aimed to provide heterogeneous analytical tools, to which not all participants need to pay equal attention. Lakin also believed that the availability of a general-purpose text-graphic editor, together with multiple special-purpose analysis tools will enable the team to switch between discussing general issues and dealing with more technical details during a meeting session.

3.2.4 Ownership: group vs. individual

There can be no groups without individuals, and this is largely true even if group members have the same background and work with common languages and tools. As a need or an obligation, an individual's identity is basic to the concept of ownership in a context of group work. A drawing created by a group member may not necessarily be *owned* by the individual, if other members are allowed to change or remove it at will. Shown by the observational studies, some researchers suggest that participants themselves are good at coordinating individual activities so that there is no need to provide extra facilities for controlling ownership; some others implicitly or explicitly address the issue of supporting the preservation of ownership control to prevent potential malicious or accidental acts such as removing individual and group work results.

Borrowing from the model of 'permissions' found in many multi-user operating systems, the levels or degrees of ownership can be defined in terms of two dimensions: *identities*, and *operations*. For identities, these can be further divided into, for example, personal, sub-group, group, all; in operations, there can be a differentiation between read, write, and execute. This file-based permission model however, can only partially

illustrate the ownership issue in a shared drawing space; a more fine-grained framework is needed.

As reported in Lu's study of teamwork in architectural design [Lu92], three user requirements were identified, revealing the need for 'seamless and dynamic' transitions between group and individual ownership:

- Allow participants to declare any portion of a sketch as private and not subject to deletion by others.
- Allow participants to identify, with no additional interaction, who owns a specific design sketch.
- Allow participants to bring in their own ideas from a private drawing surface provided by the shared tools or from a private file to a shared drawing surface.

Given the different considerations of how ownership may be defined and managed, there are currently three kinds of approaches: (1) the building of multi-user interface components, (2) the design of arbitration algorithms, and (3) ownership embedded in separate drawing spaces. To give some examples, the CaveDraw drawing surface, developed by Lu and others [LM91, Lu92], has the distinction between 'pencil' input (for producing pencil marks that can be changed by any participants), and 'marker' input (for producing marker marks in distinctive colours owned by individuals); an operation of 'cut-and-paste' is further provided to enable transitions of design ownership during collaboration[6].

In arbitrating the potential conflict of multiple users' grabbing the same *drawing object* simultaneously, the design of GroupDraw [GRWB91] regulates ownership into various levels (see Section 3.2.4. for a more detailed discussion). Being rather as a technological consequence of video-based or fused computer-video shared drawing spaces, (e.g., *VideoDraw*, *TeamWorkStation*), a participant naturally owns what he or she draws on an individual screen or desktop surface, since no one can change or erase other participants' work simply by viewing or pointing at them on one's own drawing surface. Ownership, in this approach, is inherent.

To summarise the above discussions, Table 3.2 gives an overview of the aspects of understanding shared drawing space activities. It is shown that some research aspects are comparatively less explored either empirically or conceptually than others.

[6]For instance, an individual can cut parts of a drawing originally in his or her own colour marker and then paste them into the public domain in pencil marks [LM91].

Research Aspects	Research Groups	Lakin et al.	Stefik et al.	Tang et al.	Bly et al.	Ishii et al.	Ficsher et al.	Lu et al.	Wolf et. al.	Brinck et al.
Patterns of Events	collocated & synchronous	○	●	○	○	●	○	○	●	○
	collocated & asynchronous	○	○	○	○	○	○	○	○	○
	remote & synchronous	○	○	●	●	●	○	●	○	●
	remote & asynchronous	●	○	○	○	○	●	○	○	○
Information	action-oriented	○	○	●	●	○	○	●	●	○
	representation oriented	○	●	○	○	○	●	○	○	●
Tools	homogeneous	○	○	●	●	○	●	●	●	●
	heterogeneous	●	●	○	○	●	○	○	○	○
Ownership	group	○	●	○	●	○	●	●	●	●
	individual	●	○	●	○	●	○	●	○	○

LEGEND: ● Fully Addressed ○ Not Addressed ◐ Addressed to an Extent

Table 3.2. An overview of the aspects of understanding shared drawing space activity investigated by the different research groups in the survey.

3.3 Prototype Developments and System Features

In the above, we have given a review of the current researches on the aspects of group drawing or design activities. Motivated or guided by the various understandings of *what*, CSCW researchers have worked on the problem of *how*, i.e., the development of prototype systems and the demonstration of using these tools in various contexts of group working. In this section, a survey of the system issues arising in the current prototype developments is presented. As a result of this survey, five clusters of system design issues are classified: (1) structures of graphics, (2) network configurations, (3) information storage and retrieval, (4) multi-user interfaces, and (5) other dialogue channels. The classification, which emerged from our comparative study of the prototype systems reported, is not intended to be exhaustive. Nevertheless, to our view, it is in dealing with these issues that components of experimental collaborative drawing systems were introduced. and put together. A discussion of these issues is given in the subsections below.

3.3.1 Graphics primitives and operations

Drawings as visual objects, created and passed around among people, are often constructed from some graphics primitives which may or may not have computational representations within a drawing system. In computer-based drawing surfaces, users are provided with drawing functions for making marks or constructing graphics objects such as lines, rectangles, circles etc. The provision of drawing primitives and functions determines the properties of a drawing surface to a great extent, since these primitives delimit what pictorial expressions are allowed, and what drawing operations can be applied to parts of these pictures. Certainly, the design of shared drawing spaces is not an exception to this basic principle; but the requirements for supporting possibly concurrent multi-party interaction in shared drawing space have motivated different views on what drawing primitives and operations should be provided. Five different *structures of group graphics* are found on this basic issue.

1. *video-captured images of freehand sketches.* By using markers directly on drawing surfaces, drawings are simply participants' freehand sketches. To transmit the images of sketches made by team members located in different workspaces, video monitors, video cameras, projectors, and networks are set up as working suites. Since there are no computational representations of drawings involved, participants can use white board markers to draw freely whatever they want. Supported by video networks, what appears on an individual's drawing surface is a synthesised *visual space* containing a translucent overlay of his or her sketches and video-captured images of those by others. Apart from physically erasing and taking pictures of the marks left on the drawing surfaces, little can be done on the drawings once made. Figure 3.2 to Figure 3.4 show three design examples of (purely) video-networked drawing surfaces.

2. *pixel-based graphics.* Pixel-based (or bit-mapped) graphics is often defined as the picture representation of drawings as arrays of pixels on computer screens, which corresponds closely to the data storage pattern in computer memory. As the primitive of most *painting* systems, a pixel has only the states *on* or *off*, and it has no relation to the states of others. Therefore, drawings in pixel-based graphics have typically no underlying structures or models specified by the graphics system. Drawing functions can be implemented as procedures for generating images of arbitrary marks, lines, rectangles, circles etc. Due to its simplicity, pixel-based graphics has been used in several prototypes of shared

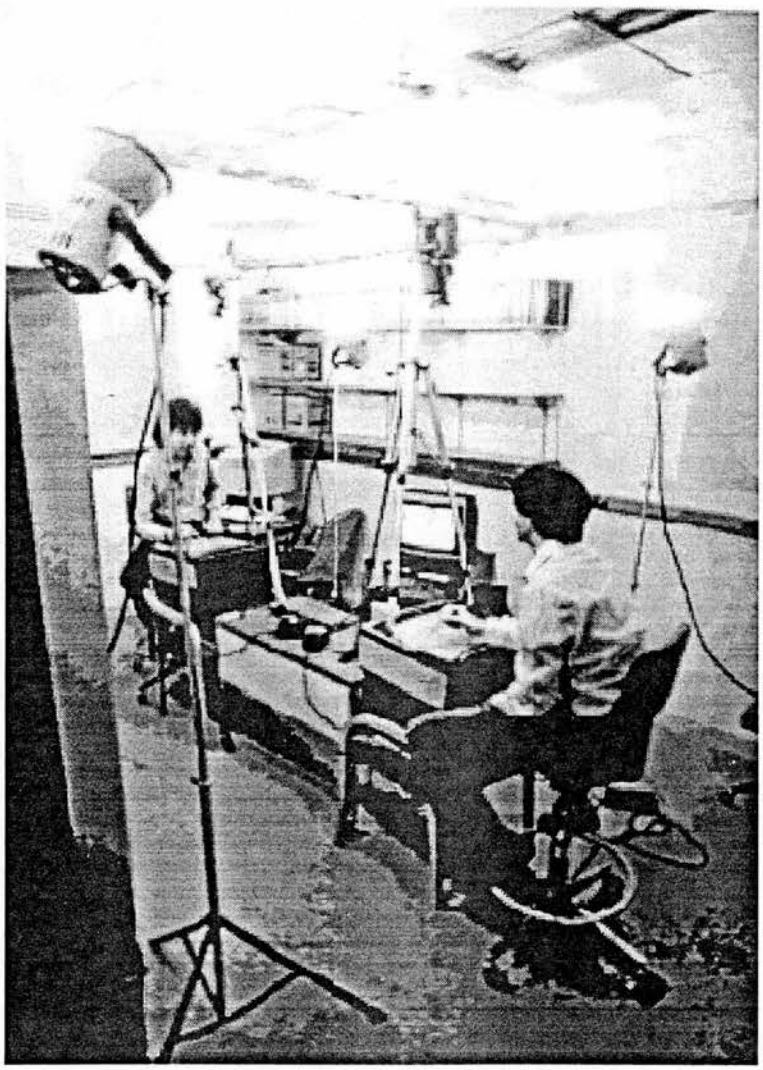


Figure 3.2. The drawing surface of *VideoDraw*, developed at Xerox PARC, used horizontal video monitor screens (20" diagonal) with dry-erase ink markers. (Source: Fig. 2 of [TM91a])

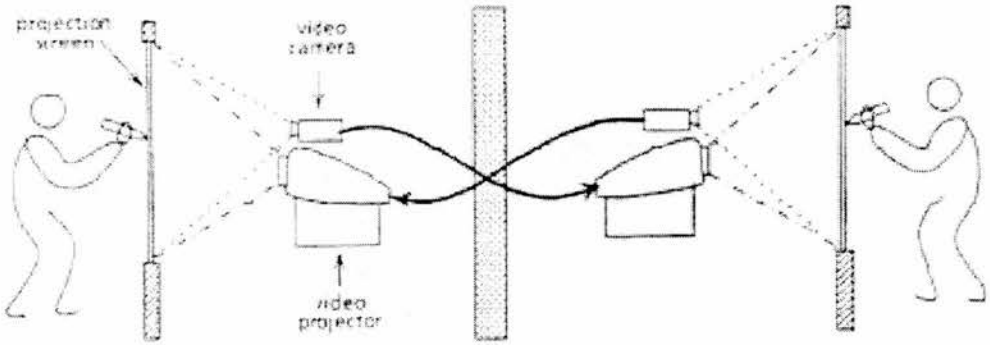


Figure 3.3. The shared drawing space of *VideoWhiteboard*, also developed at Xerox PARC, used wall-mounted rear projection screens (approximately 4.5'x6'- with standard dry-erase whiteboard markers. (Source: Fig. 5 of [TM91b])

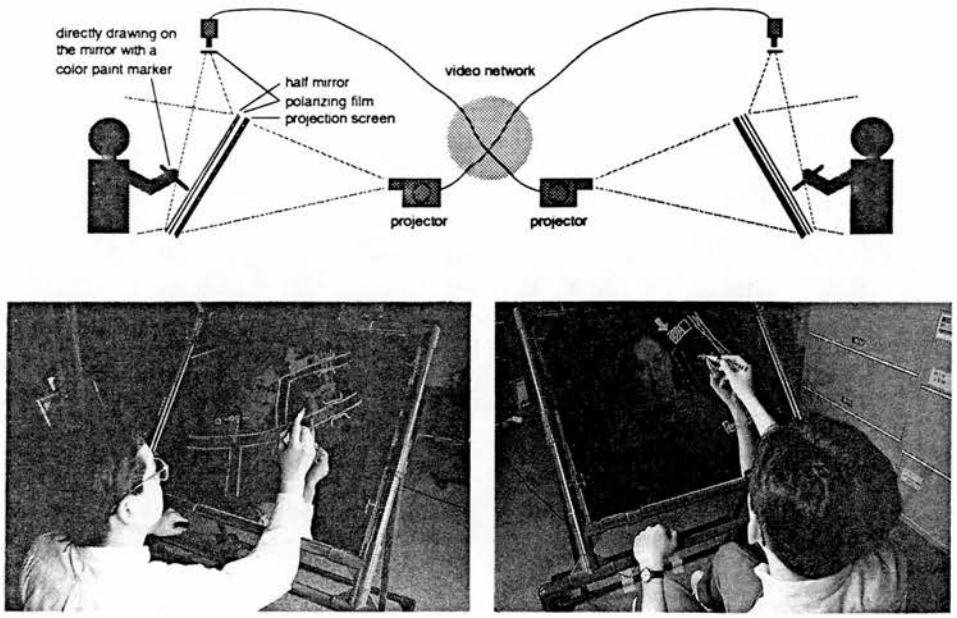


Figure 3.4. The shared drawing board of *ClearBoard-1* developed at NTT was composed of a projection screen, a polarising film and a half-silvered mirror with water-based fluorescent paint markers. (Source: A juxtaposition of Fig. 7 and Fig. 10 of [IKG92])

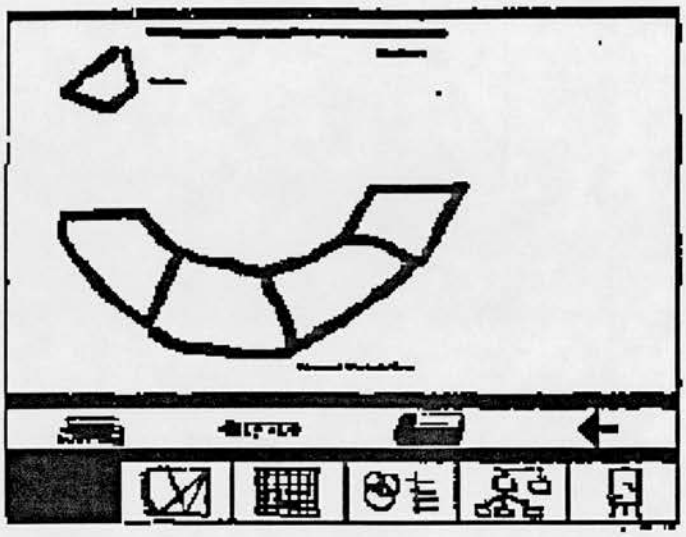


Figure 3.5. *Boardnoter* of *Colab* at Xerox PARC provides participants with mouse-driven cursors ('chalk'), operating at individual workstations but not visible on the large meeting room screen. (Source: Fig. 13.2 of [SFB+87])

drawing space. Operations like erasing, selecting-and-dragging, and cutting-and-pasting are commonly used for changing bits of drawn or textual expressions. Among the shared drawing tools built of pixel-based graphics, a diversity in the design of input devices shows different approaches to experimenting with how drawing events can be shared among participants (Figure 3.5 to Figure 3.8).

3. *object-structured graphics.* Freehand sketches and pixel-based graphics are unstructured graphical expressions to which very few operations can be applied. To be able to manipulate parts of a drawing as the constructs of line, rectangle, circle etc., geometric structures need to be included in the implementation of graphics primitives. The term "object-structured" as applied to graphics refers to a system's drawing primitives being programmed as *objects* and stored in a database to be addressed, manipulated, copied as individual entities. In an object-structured graphics system, types of drawing objects can modify themselves with various sorts of operations, such as creating, moving, resizing, grouping, rotating, duplicating, deleting etc.

   To give an example of object-structured graphics, one of the 'tool palettes' of *Conversation Board* [BG92] provides a range of geometric objects including oval,

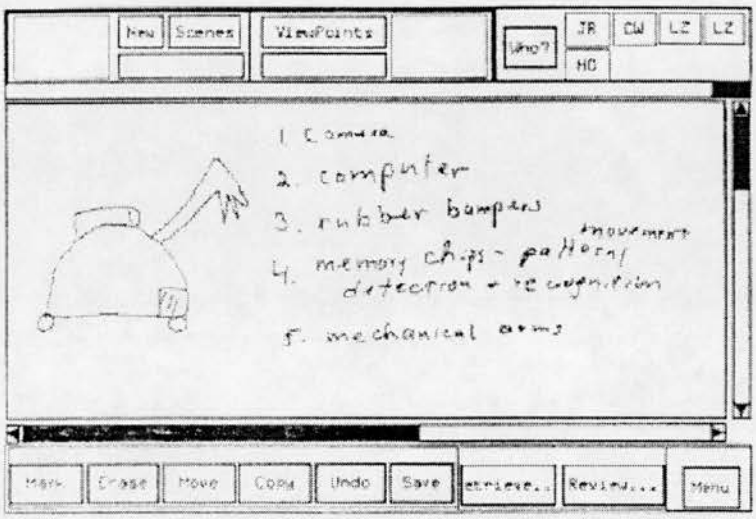


Figure 3.6. Being similar to *Boardnoter* as a meeting support tool, the *We-Met* drawing surface developed at IBM Watson Research Center has the feature of a 'pen-based' interface. (Source: Fig. 1 of [WRB92])

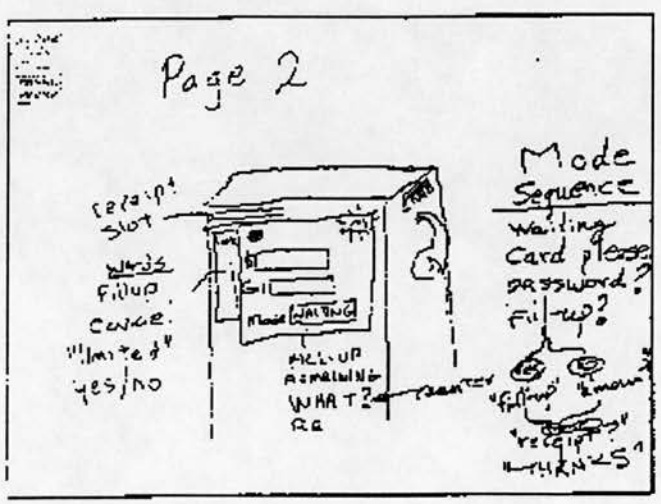


Figure 3.7. The drawing/writing surface of the *Commune* workstation is comprised of transparent digitising tablets with styli; each digitizer tablet continually reports the position of its local stylus to the processor, and each user's stylus is represented on the screen as a pencil-shaped cursor producing marks in a distinct colour. (Source: Fig. 4 of [MB91])

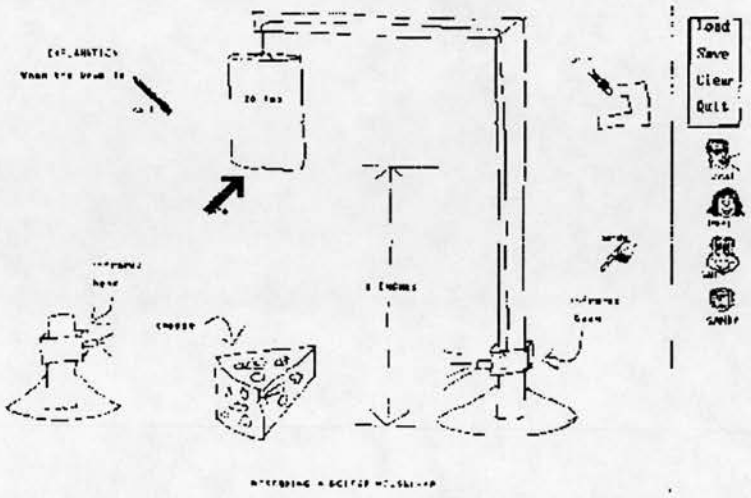


Figure 3.8. In *GroupSketch*, multiple mouse-driven cursors represented by different icons are used to convey participants' physical gestures such as drawing, typing, pointing, erasing, and directing public attention. The positions and movements of the cursors at all sites are visible to all participants in real-time. (Source: Fig. 1 of [GB90])

line, arrow, and rectangles (see Figure 3.9).

Putting object-structured graphics into group use, there arises the problem of *concurrency control* which is not significant in pixel-based group graphics. In a session of collaborative drawing, it is likely that two or more designers intend simultaneously to manipulate the same object appearing on the shared drawing surface. To coordinate users' potential concurrent manipulations, the message-sending mechanism of object-oriented programming has been used in attempting to sequence concurrent processes at different sites. A good example is the design of object-structured group graphics in *GroupDraw* [GRWB91] (see Figure 3.10).

The design of *GroupDraw* system shows an interesting attempt to integrate the design of *communication primitives* with the design of the *graphics primitives*. As Greenberg's team did, two instance variables were built into the *root object* of all graphics primitives: *ownerProcess* and *couplingStatus*[7]. By indicating who the owner of the process is, the former serves to arbitrate contention in manipulating

[7]The concepts and algorithms of *flexible coupling* and *coupling awareness* were firstly explored and used in programming multi-user interfaces by Dewan and Choudhary. See [DC91a, DC91b] for more details.

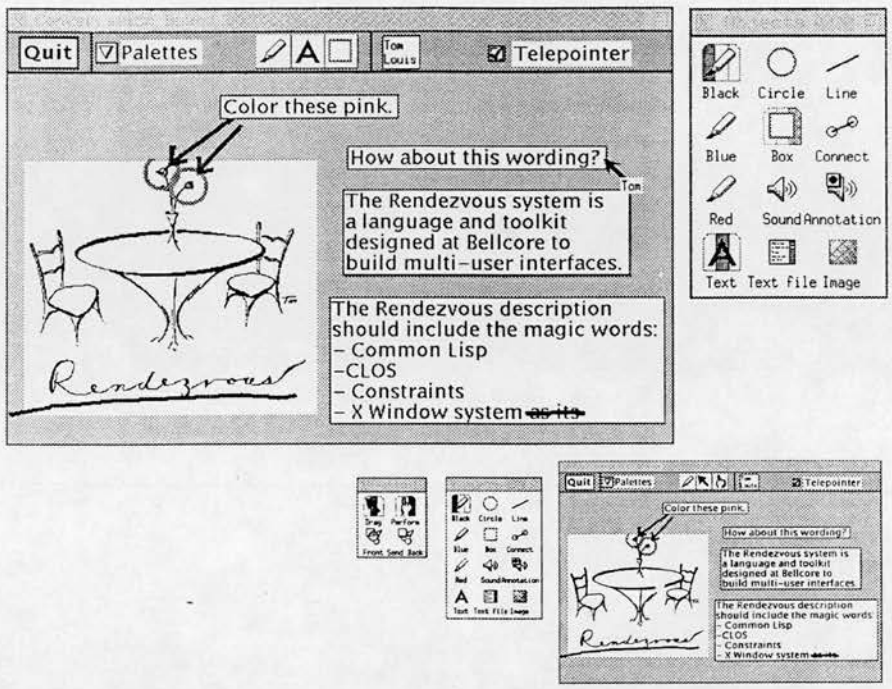


Figure 3.9. The *Conversation Board* developed at Bellcore provides a number of structured objects including oval, line, arrow and rectangle; after objects are placed on the shared canvas they can still be edited and moved. (Source: Fig. 2 of [BG92])

an instantiated object; by indicating the status of object being 'private', 'public', or 'sharable', the latter determines the extent to which graphical objects are shared [GRWB91].

4. *knowledge-based graphics.* In contrast to manipulating objects at a syntactical level (as in object-structured systems), graphical objects are defined and manipulated *semantically* in knowledge-based graphics. In a knowledge-based approach, graphics primitives are programmed in terms of abstract construction and operation components that are specific to particular design domains. Evaluations, explanations, advice, alerts, or criticism of graphical expressions constructed in those components can then be computed and presented to the designer at work. A shared drawing space connected to a knowledge base provides a set of domain-specific graphic constructs as a common design language shared by its user groups. The drawing operations, which enable a user's direct manipulations of objects, are more conceptually bound to the system's knowledge domain; for example, parts of a construction can be manipulated by changing the values of attributes

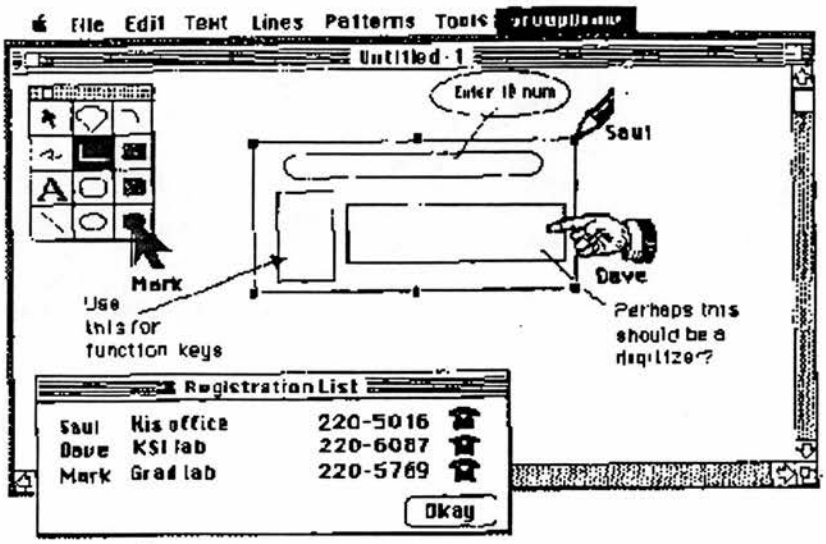


Figure 3.10. *GroupDraw* was one of the first shared drawing systems using object-structured graphics to address the issue of concurrency control in collaborative drawing space. (Source: Fig. 2 of [GRWB91])

associated with the graphical constructs embedded. At a higher level, a user may gain *multiple views* of a design by switching from one underlying construction kit to another[8]. The design of *XNETWORK* environment (a recent update of *NETWORK-HYDRA*) is such an example (see Figure 3.11), and it proposes a way of sharing drawing space through (indirect) collaborative construction of the common knowledge repository [FGL+92, RS92].

5. *semi-structured graphics.* The term 'semi-structured' refers to a picture representation resulting from a mixture of unstructured graphics (freehand or pixel-based) with structured graphics (object-structured or knowledge-based). Currently, there appear two ways of enabling the use of semi-structured graphics in shared drawing space:

[8]Note that multiple views of a *design* is different from multiple views of a *drawing*. In a kitchen design, for example, multiple views such as structure, lighting, mechanical services etc., can be involved, and each view may produce drawings in a distinct domain of construction. Multiple views of, say, a sketch of a kitchen plan, on the other hand, require multiple *interpretations* of a single graphical construction from different views. For an exposition of a multi-layered model for interpreting architectural drawings, see [Ver91].

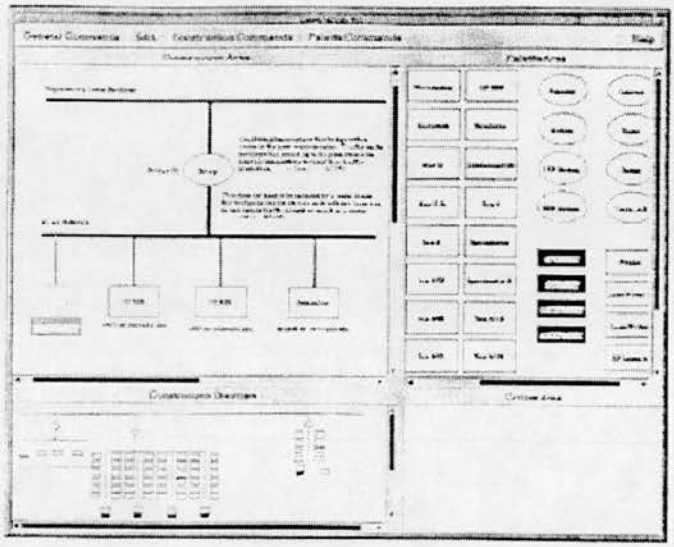


Figure 3.11. The Construction Kit of *XNETWORK* which allows for a connection between graphical construction and a knowledge base representing 'group memory' of network design. (Source: Fig. 3 of [RS92])

- Video captured images of freehand sketches *superimposed* on formal drawings generated by some graphics package — the design of *TeamWorkStation* presents a shared drawing space where transparent video images containing hand-drawn expressions are overlaid with formal drawings constructed on a computer screen [IM91] (see Figure 3.12).
- Formal drawings *embedded* in unstructured conversational sketches — the graphic editor *vmacs*[9] plus *visual languages for cooperation* is an example of this approach [Lak90] (Figure 3.13).

3.3.2 Communication networks and interprocess communications

As shown by the observational studies, the different perspectives have led to various choices of what communication networks are appropriate for supporting the various patterns of shared drawing events. The term 'communication networks' has to cover a wider scope of system architectures in implementing shared drawing space; computer networks, as usually thought of, may not necessarily be involved here. Video networks,

[9] *vmacs* is a trademark of the Performing Graphics Company.

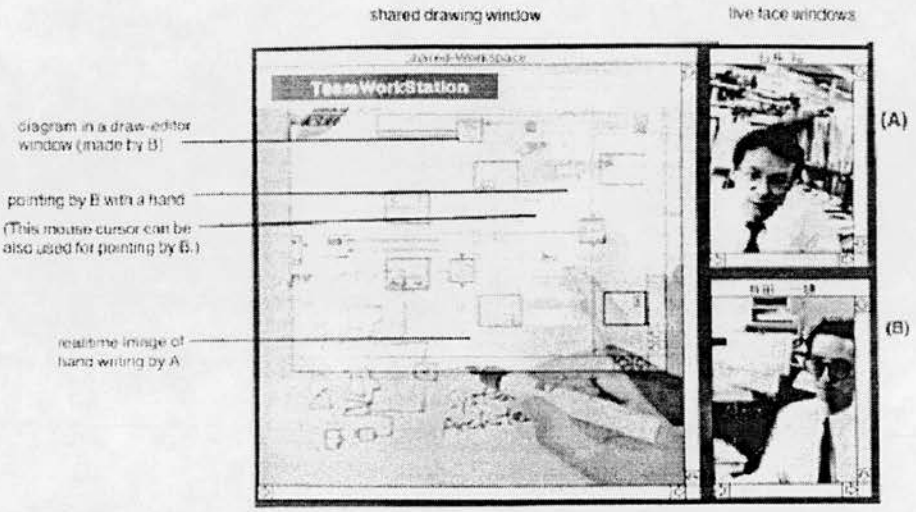


Figure 3.12. The shared drawing space of *TeamWorkStation* facilitates a translucent overlay of a desktop image of a freehand sketch and an image of operational drawing, which is displayed on the shared screen of all participants for remote meetings. (Source: Fig.5 of [IM91])

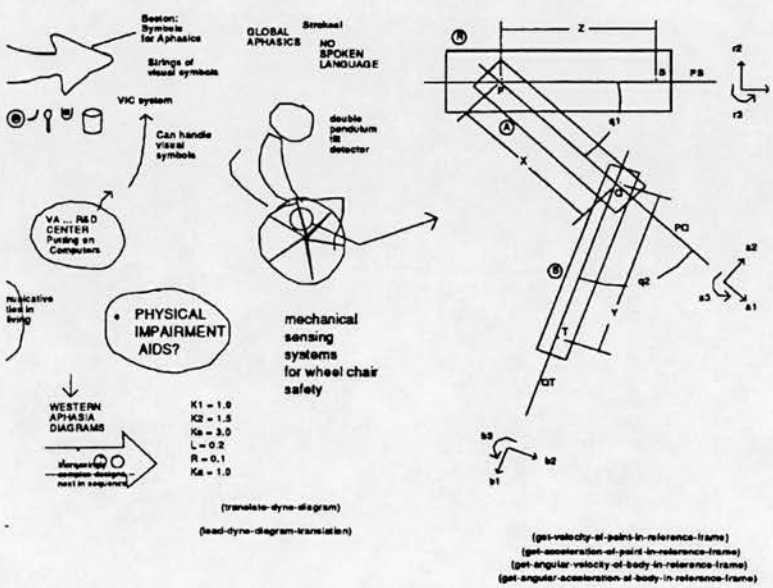


Figure 3.13. Semi-structured graphics in the shared drawing space of *vmacs* enables both unstructured conversational expressions and visual language expressions to appear in the same workspce. (Source: Fig. 17.7 of [Lak90])

for example, have been exploited to support real-time collaborative drawing sessions held between remote working sites.

When computer networks are used to serve the communication infrastructure of collaborative drawing surfaces, there arise the issues of concurrency control and maintaining consistency in data and view sharing. Message passing seems to be the most widely employed mechanism to handle interprocess communication which receives inputs from multiple users' drawing acts and delivers the computed end results to each participating site. It is possible to classify current network configurations in shared drawing space into the following four different types.

1. *hard-wired configuration.* This is the network design adopted by most video-based, or computer and video fusion approaches. The components of the network are purpose-built to perform the specific system functions as a shared drawing surface. In a fully video-based configuration, video cameras, monitors, and projectors are *hard-wired* for imaging, transmitting, and projecting the images of participants and the state of their work. There may be two main reasons why a hard-wired architecture is constructed: (a) to investigate how *tele-presence* can be realistically supported, which is conditioned by whether the configuration can convey cooperative work together with participants' 'body language' (e.g., hand gestures, facial expressions, eye contacts etc.) in the course of a remote meeting; (b) to simulate the elements of a natural setting of freehand sketching.[10]

2. *centralised configuration.* A shared drawing tool with a centralised communication structure can be explained by the 'star' network topology, in which all workstations are connected via a single link to a central switching node [SK87]. Within the configuration, a central server runs a single application and conference process. The conference process handles most of the synchronisation and serialisation issues, the application process computes output of drawing functions from input multiplexed by the conference process. Each user's workstation runs a participant process (a user interface client), providing low level interactive graphics primitives[11]. Two constructs of interprocess communication in the

[10]For example, in the prototype design of *Commune*, Bly and Minneman have described the decision of modelling the system after a shared pad of paper, which leads to the use of a horizontally positioned drawing surface and a writing tool like a pen [BM89].

[11]In Patterson's terms [Pat91] , the centralised application and conference process is the single abstraction process containing the abstraction objects for the application; and a participant process is a*view process* containing the view objects for a particular user.

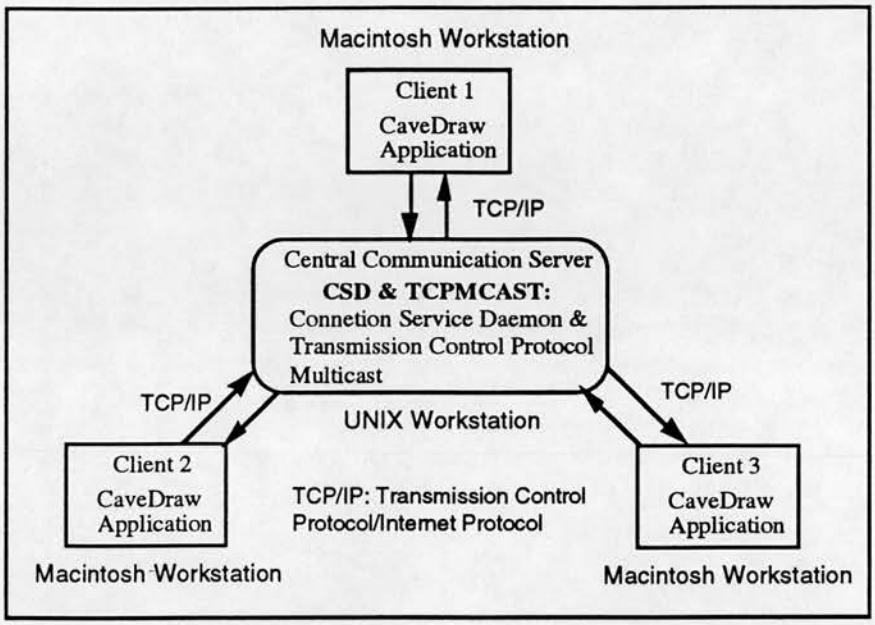


Figure 3.14. The communication structure of *CaveDraw*—an example of a hybrid network configuration. (After Fig. 17 of [Lu92])

client-server model have been borrowed from distributed programming: Rendezvous and Remote Procedure Call (RPC)[12]. The advantage of a centralised architecture is the relative ease of maintaining synchronous display among distributed participant sites. Centralised architectures may present heavy network demands if high-bandwith protocols like the X-Window system are employed in implementation, and hence they are less robust in the face of network and host machine malfunction. Another problem concerning the centralised approach is that if low bandwith protocols are used, the system may not be able to support participants' sharing the acts of creating and using drawing expressions, which is an important requirement as specified in [Bly88, Tan91].

3. *replicated configuration.* In contrast to the centralised approach, a replicated architecture runs a copy of the conferencing and application processes on every workstation that a user may interact with. The conference process at each site sends/receives input to/from other networked sites, and passes received input to

[12]More detailed explanations of the difference between the two communication primitives can be found in [BST89].

the application process for generating output and updating its resident display. Besides the merit of reduced network demands and hence gaining lower latency of application response on each local site, the replicated approach supports the conveying of drawing actions in the course of group meeting. A replicated configuration is built upon the *serial bus* or *highway* network typology[13], in which simultaneous transmission by multiple stations may result in interference; therefore, a media access control mechanism is needed to prevent or resolve contention for the transmission medium [SK87]. As a consequence, it is harder for a replicated architecture to achieve integrity of shared drawing surfaces crossing all participating sites.

4. *hybrid configuration.* Here, a participant process run on every workstation may use a central conferencing process only for serialisation and synchronisation, and all other shared drawing space acts are communicated directly between participant processes. Taking *CaveDraw* as an example, a hybrid configuration may consist of a central communication server that mediates the participating workstations which all run a copy of the application program [Lu92] (Figure 3.14). According to the *CaveDraw* experiment, there remains the problem of concurrency because of the seemingly unavoidable time discrepancy between passing local events (by any one participant process) to the central communication manager and updating local displays (by the central manager).

3.3.3 Information storage and retrieval

To complete a shared design project, it may take participants hours, days, or even years to carry out individual as well as group tasks. The third class of design issues in shared drawing space involves the facilities to record and to manage the history of collaboration. Drawings and other forms of information created and used in the past may need to be brought back to the present for individual as well as group purposes. Several notions and functions of storing and retrieving information for supporting group design activity have been attempted.

1. *camera or video record.* To make records of work results from collaborative drawing sessions, fully video-based workspaces often resort to video-taping and/or picture-taking. Since the video-based approach puts great emphasis on the sharing of drawing actions rather than on data sharing, storage and retrieval have not

[13]Note that this is true of some implementations, but is not an essential requirement—in principle, any standard network topology is acceptable in building replicated architectures.

been considered an essential function of a shared drawing surface. This decision is based on two explicit design rationales: that "drawings as artefacts in themselves are often meaningless" [Tan89]; and that "hand gestures help participants to store (to remember) design discourse" [Tan91].

2. *workstation drawing files.* Workstation-based group drawing tools can make direct use of the file management facilities of most operating systems. In supporting storing and retrieving data during collaborative drawing and design sessions, there appear to be different metaphors of storage and retrieval which lead to interfaces with various design features:

   - (time-stamped) *pages* or *scenes*: In systems like *GroupSketch*, *Commune*, *vmacs*, or *We-Met*, the interface for storage and retrieval simulates the pages of a note pad or flip chart. New 'pages' or 'scenes' can be continuously issued for making marks; and, when facilitated by a history mechanism, one or more pages can be reloaded onto the current working surface.
   - (time-stamped) *miniature sheets.* In *Boardnoter*, sketches are saved into reduced views as a collection of miniature sheets visible on the common screen. When retrieved, each sheet can be re-displayed at full screen size [SFB+87].
   - *drawing layers.* In *CaveDraw*, the use of drawing files by a team of designers comes close to that of tracing paper by a design team, which facilitates transparent overlay drafting [WP87]. Drawings sketched on one or more layers can be stored in a single file; when recalled, it can be displayed together with other resident layers on one's drawing surface.
   - *catalogue items.* In *NETWORK-HYDRA*, design representation is saved in a *catalogue* serving as a repository of designs constructed by participants over, perhaps, a long period of time. The catalogue contains several items, dealing with different aspects of the design task such as graphical construction, design rationales, design specifications etc. Existing items in the catalogue can be accessed and copied into a new construction by modifying what is retrieved; and completed instances of design can be archived into the catalogue for future use or reference [FGL+92].
   - *content-directed retrieval.* As one of the visual languages designed for cooperation, Lakin proposed a novel function of retrieving drawing files by their contents. It is suggested that content-directed retrieval might proceed by

having users write and draw their expressions in a formal visual language, such as *TEXT-GRAPHIC-QUERY*[14] [Lak90].

3.3.4 Multi-user interfaces

In supporting multi-party synchronous graphics interaction between collocated or remotely separated designers, there arise several novel design issues which are not common in traditional single-user drawing systems.

1. *telepointer.* A telepointer is a large cursor that appears on a common screen of a meeting room or on every workstation connected over a local network. As an interface device for group interaction, it is designed to be manipulated by participants (one at a time) to point to specific locations on a shared drawing surface. Telepointing is often said to simulate the conveying of information by hand gesturing in group drawing activity. Limited effects of telepointing have been reported in, for example, *Boardnoter* and *Conversation Board* (see Figure 3.5 and Figure 3.9 respectively).

2. *multiple cursors.* The rationale underlying the design of multiple cursors is that multiple *identities* can be attributed to local cursors used by individual participants. Identity of a cursor can be defined by, for instance, the colour it produces, the name of its (current) user, or the gesture indicator it serves (i.e., pen, marker, eraser, pointer, etc.). But there is a tradeoff between the support for the various modes of gesturing and the support for rapid switching among drawing, writing and other actions [Bly88, GRWB91, BG92].[15]

3. *group vs. individual views.* A shared drawing space may allow users to have different views that are local to individuals. On this issue, there appear different approaches to user-controllable view sharing:

   - A view sharing facility based on the "What You See Is What I See" (WYSIWIS) principle serves all participants' sharing strictly the same view during collaborative sessions; events taking place on any one site immediately affect

[14] *TEXT-GRAPHIC-QUERY* is a trademark of the Graphics Performing Company.

[15] The distinction between telepointers and multiple cursors made here is to point out, mainly, the different design concepts and effects resulted. As a multi-user interface device, a telepointer is more of a (generic) communication tool used to direct a single common visual focus among participants. Multiple cursors, on the other hand, are used to represent or indicate what individuals want to act (e.g., point to, draw a line, erase, write, etc.) in a session of group drawing activity.

Figure 3.15. The juxtaposition of individual screen and shared screen in *TeamWorkStation*. (Source: Fig. 5a of [IM91])

the current states of the shared drawing surfaces appeared on other sites. In this case, no individual views are allowed for private work.

- A participant's workspace consists of an *individual screen* and a *shared screen*. With this workspace setup, a user is given great flexibility of controlling what and when things are to be made individual or public. Given the view sharing design of *TeamWorkStation*, a participant can simply drag a drawing or a document image on his or her own individual screen onto the neighbouring screen which can be synchronously shared by all networked working sites (see Figure 3.15).
- The WYSIWIS principle is relaxed to some extent so that the effects of manipulating parts of a shared drawing space can be kept locally. A participant can turn away from a public domain by scrolling the shared drawing interface to different places (e.g., in *We-Met*, see Figure 3.6; and in *GroupDraw*, see Figure 3.10), or by moving onto another drawing layer (e.g., in *CaveDraw*, see Figure 3.16).

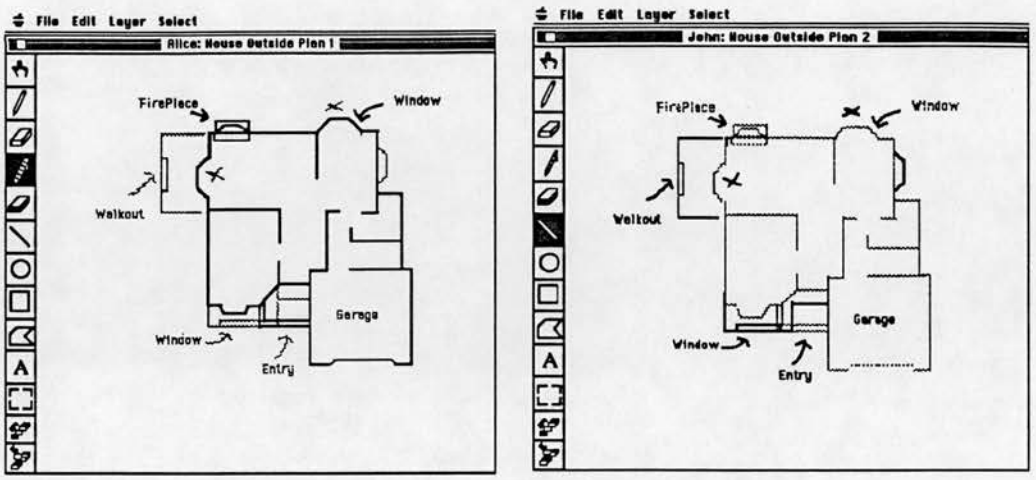


Figure 3.16. The overlapping layered approach in *CaveDraw*. Two participants can have individual drawing surfaces when co-working on different layers. (Source: Figs. 12a and 12b of [Lu92])

3.3.5 Other dialogue facilities

In supporting rich and complex collaborative design activity, communication channels other than a shared drawing surface are often provided in parallel. At least three other dialogue channels have been regularly used to complement the design of shared drawing space:

- *Audio links* are most widely employed to support participants' talking about parts of sketches while drawing or pointing at them.
- *Video links* are used not only to capture desktop images but also to transmit facial images of participants. There is a difference in the extent to which captured facial images are integrated with a shared drawing surface, and these are separate (e.g., in *VideoDraw*, *Commune*), juxtaposed (e.g., in *TeamWorkStation*), and entirely fused (e.g., in *ClearBoard-1*[16]).
- *A messaging service* is provided in connection with a shared drawing surface, enabling users to send textual or graphics messages in the course of collaboration. The MUSK system was designed with a text-based messaging service [Cra87];

[16]The design of *ClearBoard-1* was subsequently extended to *ClearBoard-2* in which a multiuser paint editor, *TeamPaint*, run on network Macintosh computers, was integrated with the original video network and drawing boards. For a detailed report on the design and use of the *ClearBoard-2* system, see [IKG92].

the visual language *VISUAL-MAIL*[17] working together with *vmacs* enables users to mail text-graphic pages [Lak90]. A mailing service is a practical alternative channel when audio/video links are not made available, or collaborative work is not based on synchronous interaction.

3.4 Shared Drawing Spaces in Three Group Uses

In the above, we have reviewed the empirical/conceptual studies of group drawing activity and the design issues in developing prototypes of shared drawing support tools. To interrelate the two parts of the survey, this section presents a discussion of the use of the prototype tools developed. Seen in this survey, it seems that the current research and development of shared drawing support tools aims to serve three different *group uses* of shared drawing tools: *conversation*, *management*, and *meeting*. Each of the three group uses reflects a way of collaborative working which points to a set of goals of system support. With this view, the current prototypes of collaborative drawing support tools can be put into three categories:

1. *as conversation media.* As seen, most of the prototype designs are concerned with supporting real-time drawing interaction among remotely located participants who, presumably, come from more or less the same professional or technical background. To be used by group members, the functions of shared drawing space are devised mainly for supporting *conversations*. Systems built for this purpose may employ graphics primitives and drawing operations ranging from freehand sketches to object-structured graphics. Since facilitating *tele-presence* is a major goal, the provision of multi-modal dialogue channels is as important as the setup of a shared drawing surface. Concerning shared use of the data resulting from conversations, however, current conversation-oriented systems provide comparatively less sufficient functionality for storing and retrieving during collaborative work sessions.

2. *as management media.* In the attempts to support collaboration via graphical and/or knowledge representations of design ideas or rationales, a shared drawing system may be characterised as a medium for users to *manage* collaborative design work. Two different views of supporting users' management behaviours have been expressed. As in the case of *CaveDraw*, one addresses the need for

[17] *VISUAL-MAIL* is a trademark of the Performing Graphics Company.

direct communication among participants; this view considers the users' need to organise design ideas that are often represented in graphical form. The other addresses indirect and long-term collaboration, considering the need of representing design knowledge in textual form; the formally captured knowledge is further connected to the system's drawing space. The shared drawing surface, as in the case of *XNETWORK*, is shared in the sense that the graphic construction space has formal (internal) communication with the state of a shared knowledge base containing design rationales written down by, perhaps, designers involved in the past.

3. *as performing media.* The third design perspective considers shared drawing space activity taking place in *meetings*, participated by two working sub-groups; namely, a single or multiple *performers* (or *demonstrators, facilitators*), and several *viewers* (or *commentators*). Group drawing or diagramming activities normally take place within a meeting room equipped with a conferencing system, as in the case of *Boardnoter*, where the performer has exclusive access to the big common screen and all other participants sitting as attendees during a brain-storming session. In *We-Met*, the strict WYSWIS feature is relaxed to the extent where participants can scroll a shared scene into private areas without affecting each other, but the change of a whole scene remains exclusively controlled by one individual. Concerning the choice of drawing primitives, the use of semi-structured graphics in *vmacs* is a novel idea, which seeks a compromise between the support of *agility* and *openness* by conversational graphics and that of expression processibility by computerised interpretations.

Based on the review of the system features and the grouping of the prototypes, an overview of the current survey is presented in Table 3.3. Given the fact that the research on collaborative drawing support tools is a continuously- (and rapidly-) expanding field, the above categorisation may be superficial. Nevertheless, to make a small contribution to the research in this field, it is hoped that this survey can serve, at least, as an index of what design issues have been considered.

3.5 Conclusions and Issues for Further Investigation

In this chapter, drawing on a selection of observational studies and prototype developments, we have presented a survey of the current trends of CSCW researches and designs in collaborative drawing support tools. This is a rapidly developing research

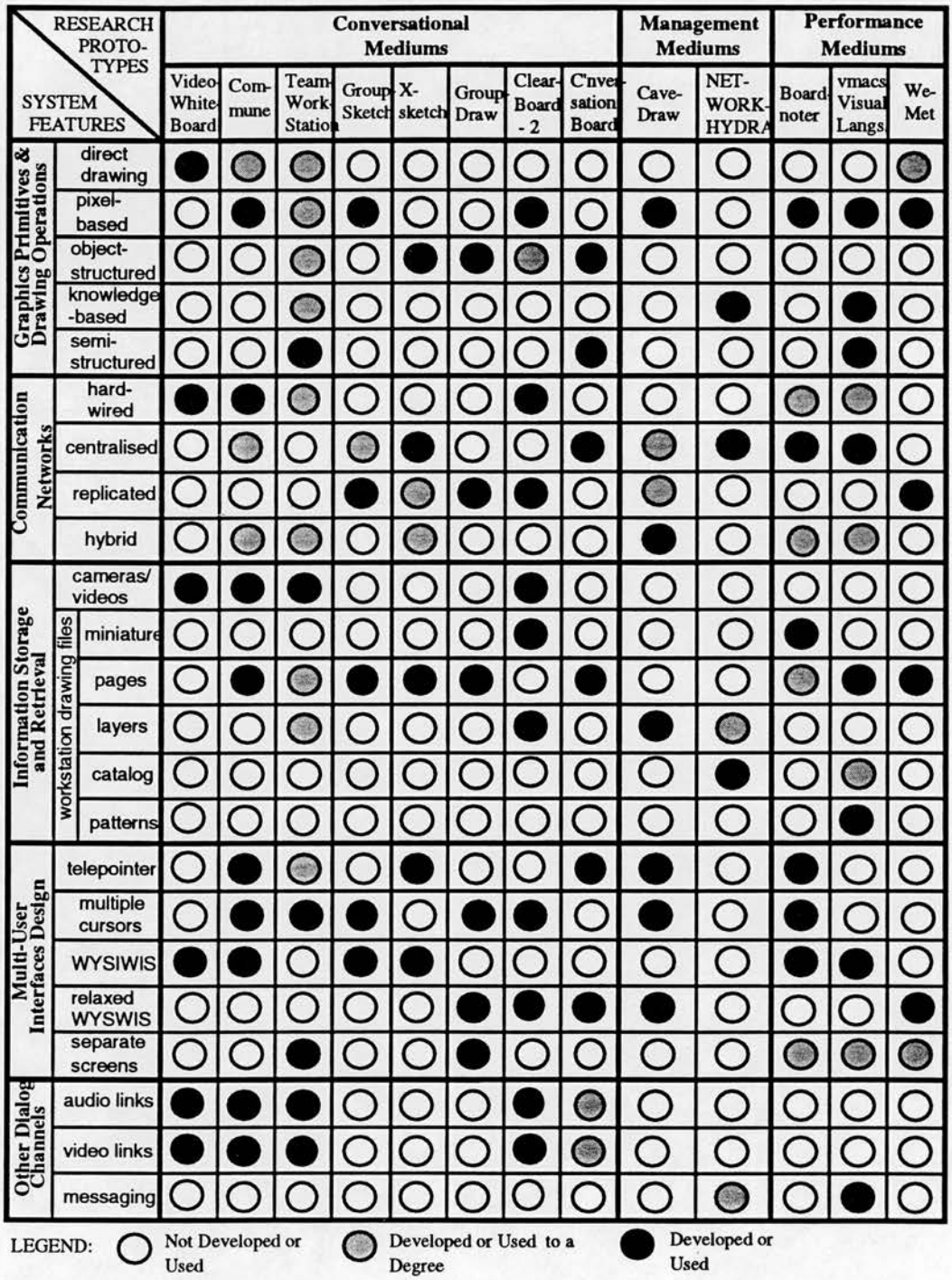


Table 3.3. A categorisation of the current research on collaborative drawing support tools in terms of different group uses and the system features.

area; during the writing of this chapter fresh work will undoubtedly have been done. But based on what we have studied, some remarks can be drawn on the nature of researching and developing system functions and architectures of shared drawing space. Basically, how do we justify the attempts that have been made, and what topics for further investigations can be suggested?

In our overall discussions, distinctions between *drawing* and *designing* have not been made. Based on the differences observed between simple electronic sketch pads for people to draw together, and rather sophisticated systems for cooperative designers to develop and manage design ideas structurally, we may point out that drawing and designing cover somewhat different aspects of human teamwork activity, which give rise to the diversity in the design and implementation of what is to be supported:

- Drawing can be considered as a general form of human communication in which pictures and texts are the traces left by communicative acts which took place sometime in the past. Shared perception and understanding of the traces, thus communication, is highly conditioned by the sharing of drawing actions that produce and use the physical traces.

- Designing seems to centre around some previously or currently existing, or yet to exist, *design artefacts*, which does not necessarily involve drawing activity; but in some design fields there can be no designing without drawing. Structures of design artefacts are introduced by, or emerge from, people's design thinking, which often conditions communication and collaboration among participating designers.

Systems that support group drawing activity must deal with the issues of supporting direct communications among users who may not be present face-to-face. Unstructured graphics proves to be a good choice for a shared drawing surface responsive to demands for (a) the intimacy between what is drawn and the actions of drawing, (b) the speed needed to maintain conversations, and (c) the freedom of making expressions. Structured graphics has been attempted in group drawing mainly for increasing the functionality of drawing surfaces and for concurrency control; but its real effectiveness in supporting synchronous graphics communication remains to be demonstrated[18].

[18]This can only be considered as a general observation. Some researchers may argue that structured graphics can provide flexibility and actually better speed in supporting synchronous graphic communication without sacrificing the advantages of pixel-based editors. However, as far as drawing or design is concerned, the qualities of effectiveness and flexibility of a graphics system can rarely be measured against a single objective or standard practice. Surely, structured graphics may have greater appeal to designers who work with a certain formal system of symbols or language.

Systems that support group work in design need to provide shared drawing space for construction and communication. The problem of how to integrate formal or informal representations of graphical objects and design ideas with an understanding of communication and coordination in design remains to be explored more deeply. We have seen two alternative approaches to group design support systems: structured construction with argumentation and semi-structured conversation with interpretation. Yet, regarding the former, we have not seen approaches that support group design processes based upon synchronous and heterogeneous representations and uses of design knowledge. In respect of the latter, the concept of supporting *autonomy* in addition to that of *heterogeneity* may need to be further addressed such that (a) organisationally, there is less discrepancy between individuals' making contributions and gaining benefits or satisfaction through the use of technologies, and (b) administratively, teamwork can be freed from one single managerial ambit. As a general direction, therefore, we have to consider what is essential to design practices that normally demand *interdisciplinary* participation, as well as *integrated* design products.

Drawing, being basic to design in many fields, is a natural mode of communication amnong members of a design team, and the research pursued in the design and implementation of shared drawing space has indeed been world-wide. It can be said that the understanding and developments made, or yet to be made, in this research area can make important contributions to the advancement of CSCW systems. Though research into computer-supported collaborative drawing or design poses distinctive system design issues and technical concerns, it is useful to see how the specialism may be related to the issues raised by the CSCW research community in general. CSCW research has a general concern: the nature and aspects of supporting *group processes*. Among many others, Paul Wilson has outlined four aspects of group processes basic to the theory, practice, and design of CSCW [Wil91]. By referring to, particularly, the first two aspects suggested by Wilson, we would like to point out that research in collaborative drawing and design presents interesting issues that are worth further investigating:

1. *individual work patterns.* The design of group support tools and working practices has to take into account individual work habits and predilections. In this respect, design activities present a high degree of idiosyncrasies among participants. How can users specify personal constructs or tools as prerequisites for supporting individual work patterns? This consideration opens up potential system design issues concerning (a) participants' *deriving* individual design spaces from the design space they share, and (b) how a common design space may *emerge* and *evolve* from the interaction between individual design spaces.

2. *representation of organisational knowledge.* Research in distributed artificial intelligence has been looking at alternative strategies and schemes of representing *organisational knowledge* in structured ways (see, for example, [Smi79, Sta89, GRHL89]). Can a simple graphical approach contribute to a better management of organisational knowledge which by its nature tends to be difficult to locate, recall, and update?

The above issues, seen in this survey and in relation to a broader agenda of CSCW research are of particular interest to our enquiry into the requirements for computer support in collaborative design. Especially, these issues are closely related to our view of design as modelling complex objects explained in the previous chapter. Collaborative design seen in this perspective involves a group of designers who communicate and coordinate with each other in the processes of modelling design objects. Drawing activity will be considered as a part of designing activity which involves other kinds of activity that may be better understood and supported as a modelling activity. In the following chapters of the thesis, we shall carry out further analyses and simulations of the structuralist and the metaphorist teamwork patterns categorised in Chapter 2. It will be seen that design as modelling and collaborative design as teamwork in design modelling can bring forward a study into both representation and communication requirements that have not been fully addressed by the current research and experimentation in supporting collaborative drawing.

Summary

Aspects of shared drawing space activities were observed by a number of CSCW research groups to identify specific opportunities for developing support tools. There are four main aspects identified in this survey: the four spatio-temporal patterns of group drawing events, the exchange of action- or representation-oriented information, the uses of homogeneous or heterogeneous drawing/design tools, and the distinction between group and individual ownership. By focusing on the various findings of group drawing activities, a range of prototype tools have been experimented with by the research groups.

It is shown that there are five classes of design issues emerging from our survey of 11 prototype systems. Under each issue, a number of technical solutions have been developed and tested. First of all, the choices of drawing primitives, ranging from natural free-hand strokes to highly structured graphical construction, seem to reflect how the researchers see drawings realised in group work contexts. Among them, the approach

of *semi-structured graphics* shows some interesting results in supporting dynamics and heterogeniety ocurring in teamwork.

Collaborative drawing can take place in a geographically (and temporally) distributed manner, if each participant's drawing platform is connected through some architecture of communication networks. Given drawing primitives and a network, information produced by participants is currently stored in and retrieved from either video-tape or a workstation filing facility. Computer-based group drawing tools require some unconventional user interface designs, including a telepointer shared by all users, multiple cursors associated with each user, concurrence control over graphical objects, and the separation between private and public drawing surfaces. Most prototype tools seen in the survey are incorporated with other dialogue channels, including audio links, video links, and message sending over networked drawing spaces.

When the prototypes were experimentally put to use, we observe three categories of group use of collaborative drawing tools. Tools in the first category, with the largest number of system implementations, are able to function as media of real-time graphical conversations. Through graphical and/or knowledge representation of design ideas or rationales, tools in the second category serve as media for managing group design work. Systems in the third category are used mainly by a single performer communicating with other collaborators in a meeting room.

Chapter 4

The Emergence of Common Design Metaphors

Having surveyed the recent CSCW developments in building experimental collaborative drawing systems, in this and next three chapters we will continue to explore collaborative design on the basis of the teamwork patterns and elements identified in Chapter 2. As some notions of architectural modelling have been introduced earlier, our current task is to differentiate the natures and roles of the common elements, such as 'common images', 'domain design expressions' etc., in greater detail. In so doing, we are able to see the implications of each of the teamwork patterns, and subsequently derive requirements for computer support. The significance of the supporting issues defined at the end is twofold: one is to relate our work to what has been pursued in the CSCW field, the other is to deliver a guideline in our further exploration of collaborative design computing.

An exposition of the 'metaphorist' approach to collaborative design is first presented in this chapter. The goal is to propose a conceptual framework, explaining how the production of integrated design is interrelated with the communication and coordination among individual contributors who work in various aspects of a design project and have different design specialisms. In addition to the case studies presented in Chapter 2, this chapter starts with another instance of the metaphorist approach to collaborative design by quoting a teamwork experience reported by a design team. An issue to be focused on here is the nature and role of *common design metaphors* that are created and shared by a team of designers. We set out to account for how 'meanings of common images' are acquired and shared among team members, and how the sharing of design metaphors is related to the developments of individual design

work in a teamwork context.

The chapter proceeds with an abstract scenario of the metaphorist approach, which presents our initial grasp of the teamwork pattern in terms of its conditions and goals observed. For the analytical framework, a brief introduction to the *situation-theoretical* perspective on information follows, including its potential application in the design of interactive information systems. The scenario is then classified into the *situation types* in collaborative design. A further look into the *flow of information* among the situation types classified helps to identify a set of *constraints on collaboration.* The metaphorist constraints spell out a logic of collaborative design: the continual interaction between the inputs from local design decisions to the production of integrated design and the feedback from resultant integration to the individual courses of design developments. Following the logic found, we continue to derive some requirements for collaborative design computing.

4.1 The Sharing of Design Metaphors

In Chapter 2, we observed two main features common to the Seattle Center Fountain project and the Cummins Research and Engineering Center project, namely, members of the design groups, at some point, produced (projected) some design images (i.e., in Figure 2.3 and Figure 2.5) on the basis of individual design contributions; and these images seem to be associated with some other things or objects (i.e., 'water-scape' and 'fishbone') commonly recognisable to the team members. We characterise these features as the 'metaphorist' approach to collaborative design.

In the following, a brief description of the Domo Serakaito House project is quoted, which, we think, is another instance of the metaphorist approach to teamwork in design [Spe91, p.15]:

> "The Domo Serakaito, built in 1974, is a house christened "coelacanth" because of its (planar) shape, created within a long, heterogeneous group process: five members of the design team designed individual sections, which was allowed to show in the clear joint within the complete building. They gain unity from the image of fish."[1]

[1]For this design project, Team Zoo's original statement read as "··· What came out from almost three years of struggle was a coelacanth that crawled out from the sea. Domo Serakanto, with its gill, spine, horn, cilium, teeth, antennae and scales, it appeared in the wind and sank in the light. Domo Serakanto is a fish dreaming about architecture, an architecture dreaming about fish." [Spe91, p.32].

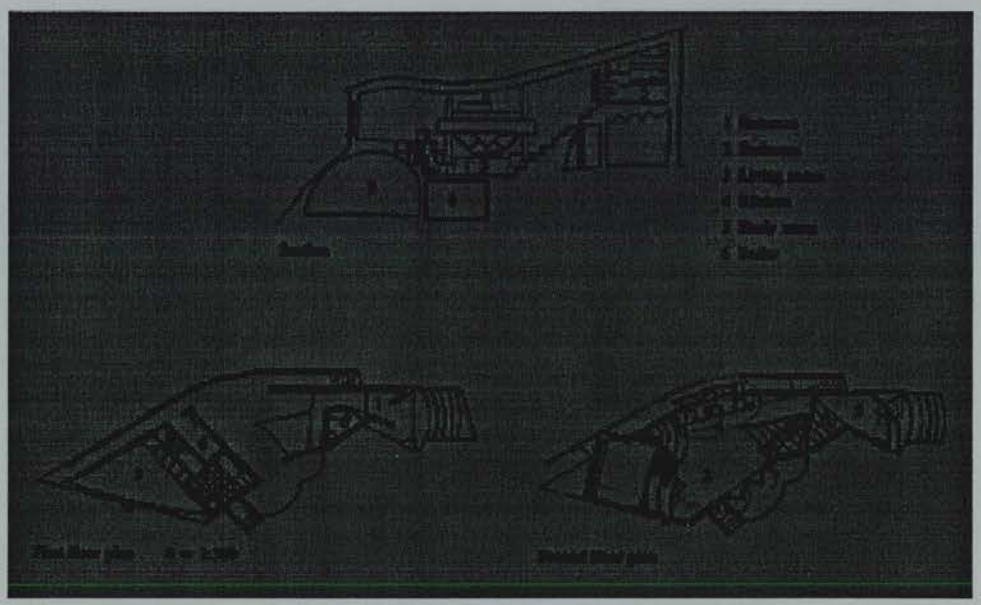


Figure 4.1. The plan of the Domo Serakanto Kamakura, designed by Team Zoo, Kanagawa Prefecture, Japan, 1974.

Rather coincidentally, the common images projected in the American engineering research centre and in the Japanese house design are all to do with the imagery of (perhaps, different species of) fish. We do not think that 'fish' is a 'standard' kind of image that is expected to result in all buildings designed by groups. It's reasonable to assume that their design results just 'happened' to be 'coelacanth' and 'fishbone' alike. However, the holistic imagery of fish gained by the Japanese design team is only part of the story; there is the aspect of, in an anatomical sense, 'parts' of the fish. That is, the *articulation* of the common design image into discrete parts that are associated with certain individuals' design work. As Spiedel put it [Spe91, p.15]:

> "Team Zoo has discovered that teamwork, even when markedly individual, can still produce a coherent whole, if a common image exists that each individual can interpret differently, and that allows a great deal of scope."

Given the instances of common images as above, we may now be more clearer about what we mean by 'metaphor' in relation to the context of collaborative design. Common design metaphor in group design can be defined as some holistic significance or meaning derived from a set of drawings (or images) viewed by a group of designers. When the viewers of the set of depictions are themselves the creators of parts of the

depictions, they may or may not reach a common recognition of a whole out of a sum of their individual contributions. In cases where a common recognition of a holistics is reached at some point, we say that the group members are 'sharing' some common design metaphor.

From the case evidence, we also know that the sharing of design metaphor is a continuous evolving process. When a whole is reached at some group design stage, it, in turn, may provide the designers with a new conceptual basis to *articulate* parts within the emerging whole (e.g., the gill, spine, horn, cilium, teeth, antennae and scales within the 'coelacanth'). Any articulation thus made may invite individual members to generate new parts, which will consequently lead to a new state of whole.

Metaphor and its use is an interesting subject for which linguists have given empirical as well as theoretical analyses (see, for example, [Lak80, MN91] among others). But there are some differences between the metaphors studied by the linguists and the sort of metaphors we propose for collaborative design. The former are largely existing linguistic entities used by people who are having conversations[2]. This is less straightforward in design. As shown, the metaphors shared among group designers are essentially graphical. Though there my be some linguistic terms associated with (e.g., 'fishbone') a design metaphor is distinctive and specific in an imagistic way.

It is not too difficult to imagine that, for different design teams, even if they reach the same metaphor named, say, as specific as 'coelacanth', it is likely that they actually produce different design metaphors in terms of the different parts that constitute the fish and how the parts are connected. Also, there can be *nameless* metaphors emerging from group design. Recall the Seattle fountain project discussed in Chapter 2, we do not see a name associated with the series of squiggles that represents the resultant fountain effects.

However, we do not feel that this is the right place to develop a deeper discussion of the distinction between graphical and linguistic metaphor. After all, the sharing of common design metaphor is only one aspect of the metaphorist pattern of group design. Our aim is to take the pattern as a whole, not to leave out other aspects without proper considerations. In the next section, an abstract scenario of the metaphorist approach is introduced. Given the scenario established, in Section 4.3, a situation-theoretical analysis is presented, showing the constitutive *situation-types* in collaborative design. In Section 4.5, a further look into the *flow of information* among the situation types classified gives rise to a set of *constraints on collaboration.* Following the constraints

[2]There must be writers who invent new metaphors when creating a new piece of literature, but this seems not the same as people's employing metaphors for communication purposes.

identified, a discussion of the issues in supporting collaborative design is given in Section 4.6. The chapter is rounded off with concluding remarks and a plan for further studies.

4.2 The Metaphorist Scenario: An Abstract

Without certain abstraction, it is not obvious where to locate a starting point for researching into the group dynamism as complex as collaborative design. This section presents an abstract scenario of the metaphorist approach to teamwork in building design. The abstract is basically an intuitive grasp of the main ingredients of the teamwork activity. A more detailed exposition of the notions outlined within the abstract will be followed in the next section.

> At the inception of a design project, designers of different schools of expertise and perspectives firstly set up individual workspaces for generating and modifying design decisions targeted at particular design aspects (domains). Participants' setting up individual workspaces may be distributed over several remote working sites without communication from the outset.
>
> At some (later) stage, participants are motivated by themselves or others to jointly present their local design decisions in a common workspace. To realise the potential connections among individual design works that are perceived during the joint presentation, group members embark on some design integration tasks in the common workspace. Examples of integration tasks are:
>
> - *aggregation* — achieving design composition on the basis of putting together spatially parts of the design expressions made by individuals into larger wholes; or
> - *projection* — achieving design composition on the basis of projecting overall design consequences by combining functionally the design properties modelled in the individual design expressions.
>
> Given the resultant integrated design, participants may be further individually motivated to carry out more design developments in one's own workspace, that is, modifying or refining their domain design decisions previously made.

> As domain design solutions are explored or elaborated to a certain extent, participants meet again. Regarding the individual design works newly arrived at, designers carry out yet another round of integration, producing newer states of integrated design.

The above description is intended as an abstract scenario of the metaphorist pattern; by 'abstract', we mean an intuitive grasp of the elements or constituents of teamwork that are common to the cases observed. The scenario given above is short indeed but sufficient to raise some questions:

- How can common design images be aggregated or projected collectively by participants if they know little about each other's design expertise?
- It seems to be the case that aggregated design parts or projected overall design consequences may give rise to new developments in individual design work, and vice versa; how can we give an account of the apparently dual communication between what is integrated in a common workspace and what is currently developed in distributed individual workspaces?
- Given the sharing of common design images among participants, how do design changes made in one individual workspace affect those in others?

Motivated by the above questions, we develop an exposition of the metaphorist pattern of collaborative design based on a situation-theoretical framework. The framework is also applied in our further investigation of the structuralist pattern in Chapter 6. As we intend to use it as a descriptive framework, a brief introduction to the situation theory in given in the next section.

4.3 A Situation-Theoretical Framework

In the follow-up of our case studies in Chapter 2, we have drawn up some general notions of the group dynamics in collaborative design. These include modelling spaces and acts, categories of design representations, and communicative acts. Based on these notions, we propose a distinction between the two teamwork patterns. In developing more elaborate accounts of what is going on in each of the teamwork patterns, we feel the need for a descriptive framework that can be used to organise the explanatory contents arrived at into a more coherent exposition. For this purpose, we choose to work with the framework provided by the situation theory originally founded by Barwise and

Perry [BP83]. In what follows is a brief introduction of the basic ideas and schemes that have been set forth by some situation theorists. On top of these ideas, we then explain how the descriptive framework can be related to our domain of concern.

4.3.1 Basic ideas from the situation theory

A comprehensive introduction to the theory of situations developed by Barwise and others is not considered necessary here. Instead, it would be more useful to introduce some of the basic building blocks of the theory that are most relevant to our enquiry in collaborative design. Furthermore, situation theory was originally intended as a mathematical foundation for formulating a formal semantics of natural language. We will not get into the detailed mathematical underpinning of these ideas, but rather attempt to share the intuitive insights spelled out by these ideas[3]

situations — A *situation* is a structured (limited) part of the world (concrete or abstract) discriminated (or individuated) by an agent. Information is always taken to be information *about* some situation, and we can have a proposition saying that some information is 'made factual by' some situation, or, to put it another way, that some information is true of some situation.

situation types — A *situation type* is an abstract (mathematical) object (in situation semantics) to represent real situations. Two unique situations belong to the same type if some 'type abstraction' can be applied onto the two situations[4]

information flow — One situation can contain or carry information about another situation only if there is a systematic relation that holds between the situation types that the two situations belong to respectively[5] This corresponds to Dretske's "Xerox principle" [Dre81, p. 57]:

> IF A carries the information that B, and B carries the information that C, then A carries the information that C.

[3]It should be said that situation theory remains a theory of information that is under development by many researchers, especially, in the formalisation of its various aspects. The basic ideas discussed here are drawn from the books written by Barwise and Perry [BP83, Bar89] and Devlin [Dev91].

[4]To give an example, the situation where *"Mary was running in Hyde Park at 3:30pm"*, and the situation where *"John was running in Princes Street at 5:00pm"* belong to the same type of situation in which "someone is running in some location and at some time".

[5]To borrow the example of information flow given in [BP83], we may think of a life sketch where Jane has a dog, Mori, who was injured in an accident. Jane later brought Mori to the vet, Fred. Fred took an X-ray picture of Mori and saw a bone fracture in the picture. Jane was then told by Fred that Mori had a fracture in her left leg. The example shows that the information about Mori's broken left leg flows from the situation where Mori was injured to the situation where Jane was aware of the fact that Mori had a broken left leg.

constraints — Constraints are the systematic relations between types of situations that allow one situation to contain information about another situation. To quote Barwise and Perry's original thought on constraints [BP83, p. 51]:

> "... for reality to support intelligent life it must be highly structured. What happens at one place and time *must* contain information about what has happened or will happen, elsewhere and elsewhen. So we need to provide an apparatus in the theory of situations to characterize this structure."

An agent's acquisition of information from a situation is circumscribed by those *constraints* of which the agent is aware, or to which the agent is attuned[6]. Situation theory characterises constraints by introducing a primitive relation between types of situations, the relation of *involving*. A constraint $\mathcal{C}$ can be expressed by a situation $\mathcal{S}$ involving another situation $\mathcal{S}'$, by writing

$$\mathcal{C} : \mathcal{S} \Longrightarrow \mathcal{S}'$$

logic of activity — The theoretical implication of the situation theory is that it provides an ontological foundation for analysing any natural activity (e.g., linguistic communication, mathematical reasoning, visualisation etc.) in the world of human being. That is, given the tools of situation theory, we can describe the internal structure (i.e., the logic) of a natural activity. The logic is a *logic of information*, concerning the constraints on the flow of information. As Barwise put it [Bar89, p. 52]:

> "When we search for the logic of some activity, what we are after is the collection of constraints $\mathcal{S} \longrightarrow \mathcal{S}'$ that govern this activity. For example, the logic of perception consists of the set of constraints that govern perception."

4.3.2 A descriptive framework for collaborative design

There are several reasons why we think that an application of the situation-theoretical framework briefly introduced above in collaborative design is appropriate and useful:

[6] Think of the example where a person acquires the age of a (dead) tree by inspecting the tree trunk left on the ground. There are the representation (the tree stump), and the item of information (the tree is, say, 45 years old). It is because he is attuned to the constraint "*the age of a tree* **is equal to** *the number of rings on the tree stump*" that the person acquires that item of information.

- Collaborative design is a natural human activity involving multiple (intelligent) agents. In principle, like any other natural activity, group design can have a situation-theoretical treatment which will lead to a search for the constraints that govern collaborative design; and this seems to correspond to what we expect.
- We like the constructs of 'situation types', 'information flow' and 'constraints', which can provide us a kind of 'quasi-formal' framework to describe aspects of group interaction in the teamwork patterns characterised before with some rigour.
- There are precedent applications of situation theory in various areas that are close to ours.[7]

However, to make our application more meaningful, it is considered necessary to fill some contexts specific to our domain of enquiry into the general framework. We therefore produce our own version of a situation-theoretical framework to be used in classifying group design activity. We start with a definition of situation types in collaborative design.

Situation types in collaborative design. In the context of group design established in Chapter 2, we shall specify three parameters (or, 'uniformities', 'indeterminates', in situation theory) in the construction of situation types in collaborative design. In our view, any situation of group design can be characterised by three elements: *modelling space*, *modelling act*, and *design state*.

modelling space — An agent (or, a number of agents) defines some *design constructs* in some medium, an abstract system, or even an existing modelling space with which a design expression can be constructed and manipulated. In the case of a single agent, we have *individual modelling space*, otherwise, *group modelling space*. The collection of constructs define the scope of a modelling space.

modelling act — An agent (or, many agents) takes some action in the modelling space that (s)he or they have defined. The kinds of actions are to do with, for example, representing, constructing, transforming, or mapping etc., of constructs or

[7]As far as we know, in enterprise integration (EI) modelling, Menzel and others argue that model integration should be based on the notion of information flow not of translation [MMS92]; in analysing and describing the fundamental social structures that influence human communication behaviours, Devlin uses situation theory as a descriptive framework and develops what he calls an 'endogenous logic' for sociologists [Dev92a]; using tools from situation theory, Devlin and Rosenberg analyse and describe how speaker and listener cooperate to achieve shared understanding in the course of communicative, natural language interaction [DR93]; in [Dev92b], Devlin outlines a perspective that situation theory can be a potential framework for the design of interactive information systems.

instances of design expressions.

design state — Whatever modelling actions taken by some agent individually or by a group of agents collectively will lead to a design state. A design state is, so to speak, an *information carrier* which contains information that may or may not have an overt representation. However, design information carried by a design state can be known to an agent or among agents through communication channels other than visual ones. Examples of design states are some collection defined by some individual in his or her own modelling space, some designer's list of questions to be answered (regarding a piece of drawing, for instance), or some design effect known to a group of individuals. We have a method to generate a range of design states for each teamwork scenario; we construct *space-action* matrices; In a sense, the matrices constructed force us to somehow exhaust the types of information carriers encountered in the group design activity.

By putting the above three parameters into a triple of [*action*, *space*, *state*], we may formulate a definition of situation types more specific to the context of (group) design as follows:

Situation Types := **Modelling Acts** *located in* **Modelling Spaces** *leading to* **Design States**

Information flow among situation types. According to situation theory, if there are systematic relations between two situation types, then information carried by situations of the types can flow from one to another. In our case, we will focus on the information flow among situation types in (group) design as defined above. Especially, we are concerned with the flow of information among situations occurring in group modelling space and those in individual modelling space. A description of information flow of this nature, we believe, will shed more light on group interaction in collaborative design.

Constraints on the flow of information. When (potential) information flow among situation types is identified, the next thing to search for is the 'systematic relations' (i.e., constraints) that allow the flow to take place. In our case, our way of finding the constraints is to infer the conditions under which one design state gives rise to another. The interrelations between the design states are relatively evident to us given our initial descriptions of the teamwork patterns.

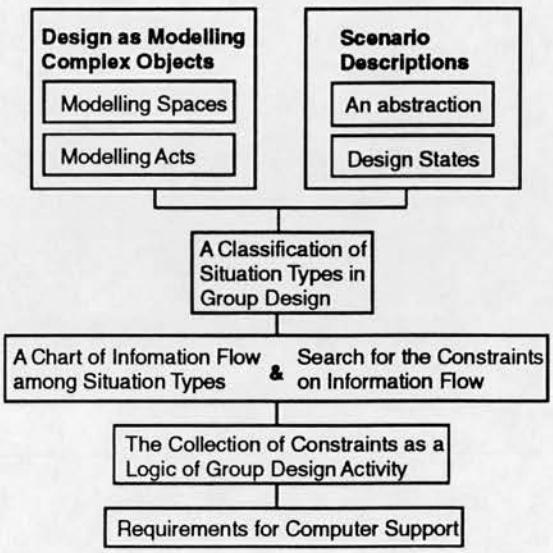


Figure 4.2. an overview of our overall approach to the analysis and description of the teamwork patterns.

Logic of collaborative design. In our situation-theoretical exposition of the teamwork patterns, we finally arrive at a collection of constraints that govern the flow of information among types of (group) design situation. According to the situation theorists, a set of constraints spell out a logic of a particular natural activity. To apply that premise to our case, we may say that we acquire the logic of collaborative design with respect to the metaphorist and structuralist constraints identified. But what is the use of the logic found? Basically, in our view, the constraints in the logic indicate where the requirements for a system to support collaborative design lie.

A summary of our overall approach to the analysis and description of the metaphorist and structurlaist teamwork patterns is provided in Figure 4.2.

4.4 An Exposition of the Metaphorist Pattern

In this section, focusing on the metaphorist scenario given earlier, a deeper analysis of teamwork in design is presented in the descriptive framework introduced above. Starting with the elements of modelling spaces and modelling acts, we record a range of situation types in the action-space matrix. More discussions are then given to each of the situation types recorded. At the end, a scheme of information flow based on the situation types classified is presented, which provides the basis for the discussion of the

constraints on metaphorist collaboration in the next section.

4.4.1 Modelling spaces

Two kinds of modelling spaces can be identified from the scenario:

IMSs — Individual Modelling Spaces (IMSs) are the kind of workspaces created and evolved by designers individually for modelling design expressions (e.g., diagrams, drawings, or any other graphical/textual constructions) targeted at a particular design aspect or domain of a design project. In teamwork, there can be many IMSs operated by participants, and by definition, an individual's creating and using his or her IMS may be physically or logically separate from others.

GMS — Group Modelling Space (GMS) is the kind of workspaces created and evolved by members of a design team jointly for modelling the integration of design parts contributed by each member into larger design wholes. A GMS is initially a *public visual space* for displaying individually made design expressions; a GMS may be later developed to accommodate new elements and functionality emerging from direct or indirect communication among participants. The emerging elements and functions are essential to the realisation of design integration as jointly intended by the group members.

4.4.2 Modelling acts

As shown in the scenario, it can be said that designers perform actions of various kinds to produce, change, or evaluate states of design works. We now look into these actions of designing more closely in the following terms:

- *Abstraction* — the acts of forming a design representation scheme[8] with which a designer establishes correspondences between his or her modelling space and the aspects of the artefact yet to be constructed in the real world.
- *Generation* — the acts of producing specific (concrete) instances of design expressions (e.g., drawings, graphical models, design specifications, etc.). In short, generation is about the use of a representation scheme by a designer's *intent*.[9]

[8]Here, a design representation scheme formed by an individual is assumed to be visible to us as observers. In reality, this may not be necessarily so in the sense that the designer has not written/drawn the scheme in any visible form but in his or her head.

[9]The notion of a designer's intent in the acts of generating design expressions should be emphasised, because a representation scheme on its own cannot motivate or explain different (specific) expressions even if different persons employ the same representation scheme.

- *Interpretation* — the acts of assigning, associating, or calibrating the values (or certain meanings) of design expressions generated. The act of interpretation often involves a designer's referring to design knowledge developed and accumulated in certain design domains (e.g., building standards, ergonomics, material strengths, etc.).

- *Modification* — the acts of making changes in (parts of) the representation schemes abstracted or the design expressions generated. The acts of modification normally have the objectives of extending the scope of a representation scheme by introducing new elements or operations, and of changing the properties and relations of design instances constructed.

As discussed before, in adding another dimension into our view of design as modelling, we may term the above design actions as 'modelling acts'. The kinds of modelling acts listed above are definitely four among many others; and there are no obvious causal relations assumed here between the acts.

4.4.3 An action-space matrix

In putting down the elements of modelling spaces and modelling acts, we have, for convenience, separated them into two camps. As read in the scenario, modelling acts always take place in group or individual modelling spaces. We now combine the two dimensions together with the aim of constructing an *action-space matrix* (Figure 4.3). The matrix is presented to generate and classify eight types of design states in the metaphorist scenario. We use the same terms for some of the design states as appearing in the abstract scenario.

Here a question that needs to be answered is—how do we know of the existence of other types of design state which are not indicated in the original scenario description? We have two reasons to make up the 'missing' items. First, a scenario is more or less an intuitive grasp of a general picture of what's going on in an activity. A scenario, like the one we present, therefore should not be considered as a 'complete' description. Therefore, what is not said should not imply the non-existence of other things. Second, for the items appearing in the matrix but not mentioned earlier, we may treat them as 'hypothetical' or 'complementary' items that wait for future verification.

4.4.4 Situation types in the metaphorist pattern

Following our situation-theoretical scheme of treating situation type as a triple of [*action, space, state*], eight situation types in the metaphorist pattern can be classified

Modelling Acts / Modelling Spaces	*Abstraction*	*Generation*	*Interpretation*	*Modification*
IMSs	*Individual Object Worlds (IOW)*	*Local Design Decisions (LDD)*	*Domain Design Agendas (DDA)*	*Changes in LDD or in IOW*
GMS	*Shared Integration Schemas (SIS)*	*Common Images (CI)*	*Common Design Metaphors (CMD)*	*Changes in CI or in SIS*

Figure 4.3. An action-space matrix generating eight types of generic design states in the background of the metaphorist pattern.

from the action-space matrix. Explanations for each item are given below.

(1) [*abs*, IMSs, $\mathcal{IOW}$] — Some designer's abstraction in an individual modelling space leads to the formation of his or her own *Individual Object Worlds*[10] ($\mathcal{IOW}$).

Basically, a designer's $\mathcal{IOW}$ consists of a collection of representation elements together with the spatial/functional operations for combining, manipulating, or transforming instances of the design elements abstracted.

(2) [*gen*, IMSs, $\mathcal{LDD}$] — Some individual's generation in an individual modelling space leads to the presentation of *Local Design Decisions* ($\mathcal{LDD}$).

Instances of $\mathcal{LDD}$ are design expressions, showing, for example, what spatial forms or particular functions are modelled by the designer when embarking on domain-specific design tasks. Depending on how an $\mathcal{IOW}$ is abstracted, an instance of $\mathcal{LDD}$ generated in an IMS may have an underlying (conceptual) structure which defines parts and part relations in the instance. In a context of group work, generations in multiple distributed IMSs may result in a number of $\mathcal{LDD}$s with, perhaps, highly heterogeneous conceptual structures.

(3) [*abs*, GMS, $\mathcal{SIS}$] — Team members' collective abstraction in a group modelling space leads to the formation of *Shared Integration Schemas* ($\mathcal{SIS}$).

This refers to the situation where group members jointly develop shared concepts and methods for the purpose of design integration. Recalling that each

[10]The term 'individual object world' is borrowed from Bucciarelli's ethnographic study of engineering design [Buc88], and is used here to denote any representation schemes formed by an individual's abstraction act performed in an IMS.

expression of an $\mathcal{LDD}$ may have an underlying conceptual structure, it is the concepts linking parts of a proposed $\mathcal{LDD}$ with those of others' that are commonly searched by participants. Several such shared concepts found can be further correlated into 'integration schemas' which can be constantly employed by the group to handle design integration at a larger scale.

To give examples of group abstraction in a GMS, consider what is needed for the integration of multiple $\mathcal{LDD}$s through aggregation or projection as mentioned earlier:

- In aggregation, instances of $\mathcal{LDD}$ made in several IMSs are *overlaid* into single representations in a GMS. Shared integration schemas for facilitating aggregation are mainly concerned with the provision of joint elements and operations that can be used to (geometrically) interrelate parts of the spatial forms or shapes that are modelled in each $\mathcal{LDD}$.
- In projection, concerns of 'functions' are greater than those of shapes. The problem is how multiple functions, as proposed by participants in their $\mathcal{LDD}$s, can be *unified* into single functions such that overall design effects can be projected in a GMS. The abstraction of shared integration schemas for projection is, therefore, mainly to do with translation and combination of different functions modelled in local design decisions.

Note that collective abstraction in a GMS to achieve shared integration schemas for aggregation or projection are merely two examples. There are other feasible ways not discussed here. As a final point concerning group abstraction in a GMS, we may ask, "Do participants always make shared concepts and methods for design integration explicit?" Lam, an experienced lighting designer, talked about 'repercussions' in project-based teamwork [Lam77, p.84]:

> "Almost every localized decision can be expected to have extensive *repercussions* on the rest of the design." (emphasised by the author).

We may infer from the above statement that if the 'repercussions' are to be kept constantly alive among participants' making local design decisions, then there is good reason for the concepts and methods jointly developed by participants to be recorded explicitly; that is, the construction of communication mechanisms via group practice in abstracting shared integration schemas in a GMS.

(4) [*gen*, GMS, $\mathcal{CI}$] — Group members' collective generation in a group modelling space leads to the composition of *Common Images*.

Given a state of $\mathcal{SIS}$ is reached by team members in a GMS, common images for the design project can be generated by putting proposed $\mathcal{LDD}$s together which are then transformed into single compositions under the operation of integration concepts and methods.

There are two general properties of common images observed in the metaphorist approach:

- *Pictorial objects.* The representation of $\mathcal{CI}$ is essentially pictorial (or, at least, diagrammatic). Given that participants know very little about each other's domain expertise, it would be difficult for them to reach a 'common sense' of teamwork without $\mathcal{CI}$ being pictorial. To a design team, instances of $\mathcal{CI}$ are generated with the property of serving all participants as *commonly perceivable* pictorial objects.
- *Composition structures.* Given $\mathcal{CI}$ being basically pictorial objects, what can we say about the structures underlying any instances of $\mathcal{CI}$? In relation to this question, two points can be discussed further:
- *Single compositions.* In the case that $\mathcal{CI}$ are generated on the basis of unified functions, conveying the overall design consequences of participating $\mathcal{LDD}$s, instances of $\mathcal{CI}$ are *single compositions* by and large. In other words, there are no abstract structures underlying $\mathcal{CI}$ that specify parts and part relations. Participants view and manipulate resultant common images in a GMS not by parts but as wholes.
- *Complex compositions.* In the case that $\mathcal{CI}$ are generated on the basis of connecting parts of an $\mathcal{LDD}$ geometrically, revealing larger wholes, instances of $\mathcal{CI}$ are *complex compositions*. That is, there are some abstract structures (captured in $\mathcal{SIS}$) specifying parts and relations among parts in an instance. It is possible for participants to view and get access to parts of a $\mathcal{CI}$ in a GMS.

(5) [*int*, GMS, $\mathcal{CDM}$] — Group members' *collective interpretation* (of the state of common image) in a group modelling space leads to the *emergence* of *Common Design Metaphors*.

Given an instance of $\mathcal{CI}$ generated in a GMS, group members search for an intersection of what the common image means to them. Instances of $\mathcal{CI}$ can be

considered as carriers of certain meanings to the participants who generate these images. We can think of two examples of how shared meanings of $\mathcal{CI}$ may arise:

- The common image as a whole *looks like* some natural objects or designed artefacts familiar to the team members; for instance, some sort of morphological or physiognomic resemblance is drawn between the $\mathcal{CI}$ currently modelled and other things.
- The common images suggest certain *ways of working* recognisable to the team members; for instance, to participants' eye, the system being developed, as manifested in the current state of $\mathcal{CI}$, works in a similar way to an existing kinematic or biological system that is commonly known to the design group.

It should be clearer now that 'design metaphors' is a notion we propose to better explain why designers of different expertise are capable of reaching shared recognition of the design images generated in a GMS. The metaphorist approach shows that group communication in design can establish through group members' sharing of common design metaphors. If communication in group design is considered not fundamentally different from that in ordinary life situation, then it is the participants' shared experiences of living in the world that contributes to their reaching a common interpretation of $\mathcal{CI}$.

However, unlike the metaphors appearing in ordinary conversations, which are, more or less, existing rhetorical devices, design metaphors need to be developed *in situ* over certain sessions of group working. Firstly, the formation of $\mathcal{CDM}$ is materially conditioned by the construction of $\mathcal{CI}$; secondly, given the uniqueness and complexity, it may well take group members a certain amount of time to resolve common interpretations of $\mathcal{CI}$. Bearing these factors in mind, it is important for us to recognise the 'emergent' nature of $\mathcal{CDM}$.

(6) [*int*, IMSs, $\mathcal{DDA}$] — A designer's *individual interpretation* (of the state of common image in his or her own modelling space) gives rise to *Domain Design Agendas.*

Given a piece of $\mathcal{CDM}$ emerging from the group processes of constructing and interpreting $\mathcal{CI}$, interpretation of $\mathcal{CI}$ may proceed at an individual level. More specifically, individual interpretation of $\mathcal{CI}$ is a designer's bringing out the significance or role of individual work in the context of an emerging whole. An individual's interpretation of $\mathcal{CI}$, being different from that of a group's, has to take the $\mathcal{LDD}$ generated at earlier design stages into account. Interpretation of

$\mathcal{CI}$ taking place in IMSs may give rise to information useful to individuals in guiding further domain design development; just to give two examples:

- A verification of the part-whole relations that have been assumed by an individual in the course of proposing original domain design solutions.
- An identification of some *potential* part-whole relations yet to be developed, regarding the discrepancies appearing between an existing $\mathcal{LDD}$ and the corresponding parts differentiated in a current scheme of $\mathcal{CDM}$.

Therefore, there is a need to introduce a special term to denote a kind of information repertoire which contains pointers to further domain design development. We propose it 'domain design agendas'.

(7) [*mod*, IMSs, $\Delta\mathcal{LDD}$ or $\Delta\mathcal{IOW}$] — Some individual's *modification* in his or her individual modelling space leads to changes in the state of local design decision or of individual object world.

Given a state of domain design development, each participant is individually motivated to modify either previous domain design decisions or, more fundamentally, the current scope or capacity of his or her individual object world. These changes can be done by an individual in an IMSs without involving others in the first instance.

(8) [*mod*, GMS, $\Delta\mathcal{CI}$ or $\Delta\mathcal{SIS}$] — Team members' *joint modification* in a group modelling space leads to changes in the state of common image or of shared integration schema.

If common images generated are of the nature of complex compositions as explained above, parts of $\mathcal{CI}$ can be decomposed and get changed. The making of these changes may be performed by some individual in a GMS, but any such changes shall be shared (or made known to) the rest of the group members. Similar situations can arise when new integration concepts or methods are abstracted to enlarge or refine the current capacity of a shared integration schema.

4.4.5 The flow of information

For every single situation type classified in the action-space matrix, we have given explanations accordingly. The next task is to 'chart' the flow of information among the types of design states explained. As already implied in the sequence of our explanations, some situations seem to follow if and only if others occur in the first instance. To see

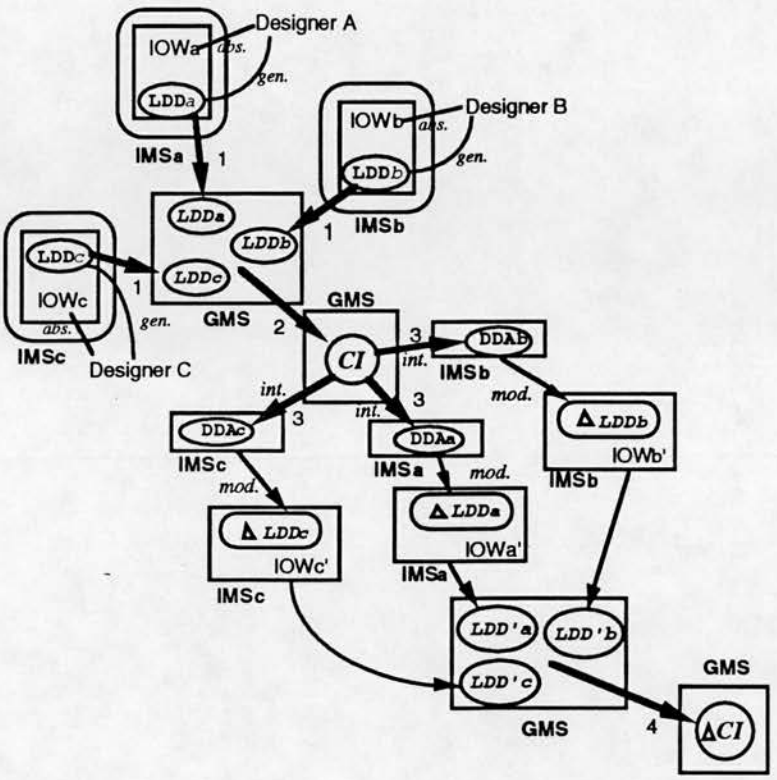


Figure 4.4. A scheme of the flow of information among the situation types classified in the metaphorist approach. (Note that the number of designers shown on this scheme is only an assumption; in theory, the number can be scaled up to any group size.)

the relations more clearly, we bring the situation types currently classified back to the metaphorist scenario given earlier. As a result, a chart of the flow of information in the metaphorist scenario is derived. To illustrate the flow, a diagram is provided in Figure 4.4.

4.5 Metaphorist Constraints on Collaboration

We now have a scheme of information flow, but how do we identify the constraints within it? If we look at Figure 4.4 again, we may find that the current scheme of info-flow contains six groupings of 'arrows' that link IMSs with GMS almost alternately. Among the six, four need to be paid more attention since some group processes are necessarily involved in these links (see the arrows annotated with digits from 1 to 4).

In this section, we will examine each of the four links and describe the conditions that allow the links to occur. Our way of searching for the constraints is basically discussion-based.

(1) from *distributed local design decisions* ($\mathcal{LDD}_a$, $\mathcal{LDD}_b$ $\cdots$) to a *collective presentation of* $\mathcal{LDD}$ (denoted as Σ $\mathcal{LDD}_i$) — i.e., from a situation (type) where multiple local design decisions distributed over a number of IMSs to a situation where the distributed local design decisions are commonly present in a GMS.

Question: What is involved when members of a design team decide to jointly present their local design decisions to one another?

The transition suggests the following conditions to be fulfilled:

- A *call for participation* is sent out by a member (or members) to other members of a design team, which specifies when and where a design meeting will be held;
- In response to the call, team members turn up for the meeting and bring along their latest design developments in various design domains;
- A *common visual space* is set up for displaying all participants' $\mathcal{LDD}$s such that potential relations or connections among parts of the $\mathcal{LDD}$s can be perceived and discussed among the participants.

The above three conditions point to the need of holding *design meetings*, in which design decisions made by different individuals in distributed sites are gathered together in a single workspace. Regarding this, we may formulate our first constraint on collaborative design in the following format[11]:

$$Constraint\ 1 : ([\mathcal{LDD}_a, \mathcal{LDD}_b, \cdots] \rightsquigarrow \Sigma\mathcal{LDD}_i) \Longrightarrow Meeting$$

(2) from Σ $\mathcal{LDD}_i$ to *the generation of common images* ($\mathcal{CI}$) — i.e., from a situation where local design decisions are present in a GMS to a situation where common images are generated in the GMS.

Question: Given a common visual space is available, what is involved in participants' arriving at common design images from their joint display of local design decisions?

[11] A general reading of all the terms and notations employed here in describing the metaphorist constraints on collaborative design is provided in the Glossary—see Appendix A.1, page 207, and Appendix A.3, page 209.

When participants of a design meeting envisage that parts of their local design decisions can be interrelated in one way or another, common design images can be constructed in a shared workspace on the basis of gathered $\mathcal{LDD}$s. To realise the relations intended, the generation of $\mathcal{CI}$ may necessarily involve sets of *concepts for integration*, for example:

- concepts in terms of *new design elements* that can be deployed by participants when joining their local designs into a larger composition;
- concepts in terms of *spatial operations* that can be applied to geometrically transform parts of local designs prior to the final generation of integrated design images;
- concepts in terms of *projective operations* applicable to project certain kinds of overall design effect on the basis of multiple inputs of local design decisions.

As described in our action-space matrix, *shared integration schemas* may result from group members' joint abstraction of the means for realising the intended interrelations among individual works. Seen in the info-flow framework, $\mathcal{SIS}$ has a functional role to play — elements of $\mathcal{SIS}$ constrain the flow of design information from gathered $\mathcal{LDD}$s to $\mathcal{CI}$.[12] In this regard, the second constraint on collaborative design can be denoted as follows:

$$\textit{Constraint } 2: \ (\Sigma\mathcal{LDD}_i \rightsquigarrow \mathcal{CI}) \Longrightarrow \mathcal{SIS}$$

(3) from $\mathcal{CI}$ to *distributed domain design agendas* ($\mathcal{DDA}_i$) — i.e., from a situation where common images are generated in a GMS to a situation where participants acquire domain design agendas.

Question: How do participants acquire their domain design agendas for further design developments in relation to the common images generated previously?

There are evidences from designers' restrospections showing that a state of $\mathcal{CI}$ can be mapped onto (interpreted as) particular *templates* or *patterns* which serve members of a design team to achieve a sense of *wholeness*. The mapping (or interpretation) may be initially proposed by some individual(s), and then get

[12] A further analysis into what is involved in participants' developing shared integration schemas is presented in the next chapter.

recognised by the rest of the team members.[13] When such an interpretation of $\mathcal{CI}$ is commonly agreed by all participants, we say that a common design metaphor has emerged.

Here is a situation similar to the use of metaphors in ordinary conversations — the creative uses of some rhetorical devices for the purpose of communication. However, as mentioned before, the metaphors created and passed around by designers are more of the nature of imagery or graphics, hence the term 'design metaphors' is introduced.

Given a metaphorical framework emerging from group design processes, participants can always have individual interpretations of $\mathcal{CI}$ with a different purpose from that of inter-personal design communication[14]. An individual's interpretation of $\mathcal{CI}$ is mainly oriented toward a re-formulation of existing relations (or an articulation of new relations) between individual contributions and emerging wholes.

This is like a participant's reflecting on (1) what his or her (new) local role is about, given an emerging design context rendered in $\mathcal{CDM}$; (2) what needs to be done to fulfil the role more exactly. Therefore, interpretation of this mode involves a domain- specific design perspective and knowledge that an individual is working with. A domain design agenda may arise from an individual's design enquiry of this kind. To different participants, the agendas acquired highlight design issues to be handled in refining or changing domain design decisions previously made.

Following the above account, we may point out that without the emergence of $\mathcal{CDM}$, it is less likely that each design party could draw up instant guiding agendas that are pertinent to continual domain design developments:

$$\textit{Constraint } 3: \ (\mathcal{CI} \rightsquigarrow [\mathcal{DDA}_a, \mathcal{DDA}_b, \cdots]) \Longrightarrow \mathcal{CDM}$$

(4) from *design changes in local design decisions* ($\Delta\Sigma\ \mathcal{LDD}_i$) to *design changes in common images* ($\Delta\ \mathcal{CI}$) — i.e., from a situation where some design changes in

[13] A deeper exposition of what constrains a person's ability to suggest such an interpretation and the abilities of others to recognise the proposed interpretation is beyond the scope of this thesis.

[14] It may be questionable if all participants can always have individual interpretation of a state of $\mathcal{CI}$. As an observer, we may have difficulty in knowing what an instance of metaphor is, since it refers to something outside any currently realised GMS, so it is unclear to us how it constrains or restricts an individual interpretation of a currently realised $\mathcal{CI}$. We can only comment on the general purpose of a designer's interpretation of $\mathcal{CI}$ when working on an individual base.

local design decisions are intended and expressed by one or several participants to a situation where participants come to know the consequent design changes to be affected in common images.

Question: Assume that some design changes are made by some individuals (in accordance with their domain design agendas), how are the changes intended by the individuals going to be realised in connection with the changes resulting in common images?

Given a newly acquired $\mathcal{DDA}$, a participant may proceed to develop his or her domain designs further, resulting in certain (intended) changes, regarding the existing $\mathcal{LDD}$, or more fundamentally, the current status of $\mathcal{IOW}$.

Due to participants' sharing a working protocol for design integration, changes intended in one design domain may have critical implications for the works pursued in other domains; that is, the 'repercussion' effect. Therefore, for an individual to be able to actually realise his or her intended changes, they have to be publicised to other participants for holding an 'exploratory integration'. By examining the consequent changes in the current state of $\mathcal{CI}$, participants can have their own (domain- oriented) judgements for supporting or rejecting the changes proposed. More specifically, we may think of the following examples of group interaction involved in making design changes:

- *backtracking.* The proposer has to drop the intended changes because some members cannot accept the outcome or the implications of the proposed changes from their own design perspectives;
- *competing.* The proposed design changes are not agreed by some other members but invite the members' design thinking, and they may subsequently produce alternative design changes that compete with the original ones;
- *coordinating.* Participants accept the result from an explorative integration and respond to the changes projected by making changes in relevant design domains to coordinate the proposed ones;
- *confirming.* Integration results judged satisfactory to all participants, and it does not demand further changes to be made in relevant design domains; participants simply send their confirmations to the proposer(s).

Viewed as the above, communications among team members are necessarily involved in the transition from (gathered) changes in local design decisions to a

new state of common images.[15] To better summarise what conditions information flow in this section, a *Consulting* constraint is expressed as follows:

$$Constraint\ 4 : (\Sigma\Delta\mathcal{LDD}_i \rightsquigarrow \Delta\mathcal{CI}) \Longrightarrow Consulting$$

To summarise, the four constraints discussed above spell out the interior situational logic of collaborative design in the metaphorist pattern: the continual interaction between the inputs from local design decisions to the production of integrated design and the feedback from resultant integration to the individual course of design developments. The sharing of common design metaphor is essential to the continual group interaction, which, in turn, is governed by group members' joint abstraction of integration schemas for the projection of common images.

4.6 Issues in Supporting Metaphorist Collaboration

Given the current exposition of the metaphorist approach to collaborative design, it is now relatively easy for us to pinpoint where the communication and representation requirements for computer support lie. For this, we conclude the following four supporting issues.

(Issue 1) *Supporting joint presentation of local design decisions.*

What is required is basically a *common visual space* in which members of a design team can jointly present their (up to date) $\mathcal{LDD}$s so that potential connections among individual work can be envisaged by whoever participates in the meeting. In this regard, typical operations to be supported include

- *networking.* In developing design solutions with a view to satisfying domain design requirements, participants create and evolve their own modelling spaces; these individual modelling spaces can be distributed over several remote working sites, and participants may have different preferences of when to work.

  A *common visual space* should be operating synchronously or asynchronously among all participants' working sites, providing designers with immediate

[15] There lies a basic difference between the constraint of *meeting* described earlier and the constraint identified here; the meeting constraint points to the joint abstraction of means for design integration, while the latter one points to the joint judgements of the results of design integration.

access to a shared view of what design solutions have been explored in various domains.

- *meeting scheduling.* At any time, one or more participants may request the holding of a design meeting to keep up to date the interrelations among local design developments. Requests for meetings may indicate a possible time and intended contents of the proposed meeting event.

  Messages of *call for meeting* are formed and sent to remote working sites, asking for participants' replies; when replies are gathered and processed, a *scheduler* should broadcast a meeting programme to all working sites.

- *structure filtering.* Local design decisions are modelled by participants within individual object worlds that correspond to their domain expertise; therefore, the conceptual structures underlying these locally shaped expressions are likely to be heterogeneous to one another.

  When imported into the common visual space, each local design expression is a *filtered image* in the sense that the domain- specific structure is left out and substituted with some kind of primitive structure supported by the common visual space.

(Issue 2) *Supporting joint abstraction of shared integration schemas.*
Participants need to work with a *common design language*[16] with which each designer can collaborate with others on defining *emerging design elements* which can then be employed when combining (spatially or functionally) parts of $\mathcal{LDD}$s, or specifying *transformation rules* that can be activated to transform parts of $\mathcal{LDD}$s for the purpose of integration. To assist group abstraction of $\mathcal{SIS}$, the following operations need to be considered:

- co-specifying common constructs as joint elements or functions on the basis of direct use of the common design language provided in a GMS;
- co-specifying production rules as transformation procedures on the basis of translating and combining domain constructs that have been developed locally.

[16] In a superficial way, the term "common design language" refers to a general representation framework that each participant is (or, at least, can learn to be) familiar with; in theory, the framework should provide a set of "basic constructs" (geometrical or otherwise) together with a "naming" facility, but what exactly the language consists of is not clear at this stage.

(Issue 3) *Supporting joint interpretation of common images.*
Currently, we observe no clear evidence showing that common design metaphors are represented in any explicit way, but are seemingly kept in participants' heads. However, we may think of the following measures to enhance group interaction in recognising the significance of a newly generated common image:

- a *distributed database* allowing participants to quickly store and retrieve *visual references* for communication purposes;
- a *highlighting* facility allowing participants to extract or trace configurations of "wholeness" from any state of $\mathcal{CI}$.

(Issue 4) *Supporting consultation in making design changes.*
Depending on how shared integration schemas are co-specified by team members, one designer's making changes in his or her design domain may cause further changes to be coordinated by designers working in other domains. A housekeeping-like mechanism can be developed to assist consultation among participants for resolving potential conflicts as manifested in the changing states of common images. More specifically, a bookkeeping agent is expected to perform coordinating tasks below.

- Detecting state changes in $\mathcal{CI}$ arising from local changes proposed by any participating parties;
- Knowing and locating whomever needs to be consulted whenever a state change in $\mathcal{CI}$ is detected;
- Sending reports to the right participants requesting confirmations or other modes of coordination;
- Delivering all responses from members involved in the consultation to the designer(s) making the proposed changes.

Summary

This chapter sets out to give an account of the metaphorist constraints on collaborative design in accordance with a situation-theoretical framework. Given the initial abstract of the scenario, we have conducted a situation-theoretical analysis of the teamwork pattern, focusing on various modelling acts situated in individual as well as group modelling spaces. As a result, we have identified four constraints on information flow

from one situation type to another as classified. The constraints identified constitute what we think of as a logic of collaborative design—the continual interaction between *the contributions of multiple design expertise to the production of integrated design* and *the feedback from an emerging communication framework, as manifested in the changing states of common images, to the evolution of individual courses of design developments*; and *shared integration schemas* and *common design metaphors* are the keys for interaction of this nature to be continuously supported in collaborative design.

Following the constraints discussed, we then turn to a view of these constraints as pointers to further development of collaborative computing systems. Some requirements for tools to be supportive of the metaphorist teamwork pattern. This chapter proposes no specific solutions to how collaborative design may be computer-supported but reports on some basic requirements to be taken into account in future system building. Surely, there are many regions to be further articulated. In particular, a deeper understanding of the structural aspects of joint abstraction of shared integration schemas seems naturally to be the subject of the next chapter.

Chapter 5

Joint Abstraction and Use of Shared Integration Schemas: A Simulation

This chapter presents a computational simulation of the metaphorist approach to collaborative design, focusing on the issues of joint abstraction and use of *shared integration schemas*. The purpose and method of carrying out the simulation is explained in the beginning of the chapter. We then describe a scenario of collaborative design exercise where two designers work jointly to achieve integrated design objects on the basis of their individual design objects modelled in two separate conceptual domains. Prior to a presentation of the simulation, an introduction to the algebraic specification language OBJ3, which we adopt as a simulation platform for the exercise, is provided. After the brief introduction to how the language works, we report on how the scenario of collaborative design as described earlier is simulated in OBJ3. With the current simulation, we bring out some internal structures of the collaboration tasks in joint abstraction and use of shared integration schemas. As an extended study of the metaphorist requirements for computer support, the implications of the simulation for a system strategy are discussed.

5.1 More on the Constraint of $\mathcal{SIS}$

In the previous chapter, the metaphorist pattern has shown us that design comes from a continuous integration of parts that are under continual design developments pursued by members of a design team. We propose a descriptive theory of the metaphorist

approach to collaborative design, which says that participants develop individual design work in correspondence with their sharing of common design metaphors emerging from joint interpretation of common images. Common images, in turn, come from a projection or aggregation of individual design objects modelled by each group member in an individual modelling space.

We have given a situation-theoretical account of what might condition the metaphorist collaboration in design. It seems to us that, among the four constraints identified, the constraint of *shared integration schema* ($\mathcal{SIS}$, as denoted in *Constraint* 2 : $(\Sigma\mathcal{LDD}_i \rightsquigarrow \mathcal{CI}) \Longrightarrow \mathcal{SIS}$, on page 98) deserves a deeper analysis. The constraint says that the production of common images from a collection of local design decisions involves shared integration schemas.

As explained before, the emergence of common design metaphors in collaborative design is closely associated with certain images realised in some graphical form. Graphical realisations of common images, as seen in our case studies, are mostly to do with whether participants can put their individual design parts together in a way that can produce *integrated* common wholes[1]. We suggest that for any teamwork to achieve common images through putting distributed parts together it must be based on some integrative concepts or operations. In our descriptive theory of the metaphorist approach, a set of constructs and operations that can be applied to establish interrelations among parts of local design decisions is termed as 'shared integration schemas'. An integration schema is 'shared' in the sense that it has to be developed jointly by participants working in different individual object worlds.

Though we know in theory the existence of $\mathcal{SIS}$, it remains unsaid in the previous chapter what group interaction is involved in the development of a shared integration schema. Given its importance in the metaphorist approach, this chapter is dedicated to a deeper study of this particular aspect of teamwork. Our method of carrying out this study is by setting a design exercise of constructing simple shapes presumably undertaken by two imagined designers. We then represent the exercise in a computer language to simulate the situation where a common set of integrative constructs and operations is developed on the basis of the individual design worlds developed by the designers. The aim of the simulation exercise is threefold:

1. to give an example of symbolic representation of a shared integration schema;

2. to extract the *internal structures* of collaboration tasks in joint abstraction of

[1]Note that we do not suggest that there are explicit or objective standards applicable in judging if a design is an 'integrated common whole' or not. This is a matter for the design group to decide when seeing a resultant assemblage of parts.

$\mathcal{SIS}$;

3. to identify the potential areas or topics for developing system strategies if joint abstraction of $\mathcal{SIS}$ is considered as a form of collaborative design computing

As a by-product of the simulation, we find that the *use* of an existing $\mathcal{SIS}$ also presents no less important issues of collaboration than those occurring at the stage of joint abstraction.

However, there are conditions on what is to be expected from the simulation exercise. First, the scope of our current simulation is restricted to simple shape construction. There is no claimed close resemblance between our highly simplified design exercise and the building design practice in the real world. Within the scope we consider that different sets of (two-dimensional) spatial concepts or constructs can be used to mark the differences between individuals, i.e., the design participants. Shapes, produced by the individuals using their personal spatial constructs, can then motivate or evoke interactions among designers, which lead to the group process of interrelating individual shapes into common spatial structures shared by, perhaps, other team members. For this reason, a highly conceptual programming language OBJ3 is adopted as a simulation platform which allows a *modular algebraic specification* approach to specify spatial constructs and to generate instances of shapes.

Secondly, regarding design as modelling objects in computers, we have a rationale for formal (algebraic) specification of spatial constructs. Specifications of spatial constructs can be thought of as producing *shape structures* as a theory, and design is the actual use of these structures in the construction of the intended shapes from the theory[2]. Methodologically, by aiming at the computational representation of an overall process of joint shape constructions, we need to narrow down onto an analysis of the functionality of spatial concepts, as symbolically specified by different individuals, in group modelling of shapes.

Given the basic conditions of simulation described above, we propose that a further analysis and description of the constraint of $\mathcal{SIS}$ can be based on the following individual and group tasks:

1. *Individual tasks:*

   - defining (personal) spatial constructs as individual modelling methods;

[2]Here, we may draw a comparison with the model-theoretical approach to software development, where formal specification of user requirements is thought of as producing a theory which describes the computing system to be built, and design is thought of as finding the most appropriate model of the theory [WL88].

- generating shape instances in one's modelling method;

2. *Group tasks:*

   - communicating how shape instances can be related with one another by developing common operations for transforming and combining source shapes into integrated shapes as shared spatial structures;
   - coordinating in changing parts of source shapes to yield different states of the shared integrated shapes.

The rest of the chapter is organised in the following manner. In the next section, a scene of joint shape construction is first described, showing some of the ingredients to be simulated. A short introduction of how the language OBJ3 works by examples is given in the third section. After all this, our simulation of the design exercise in joint shape construction is shown in the fourth section. Finally, the implications of this study for developing system strategies of collaborative design computing are discussed.

5.2 A Scenario of Joint Shape Construction

Consider a design project consisting of two participating domains: (1) *Enclosure*, and (2) *Opening*. For the Enclosure domain, one of the participants, say, Designer A, is responsible; and A creates the modelling method, $Method_A$, which captures what is required in constructing and manipulating the elements of Enclosure as A sees it. Another participant, say, Designer B, is responsible for the modelling of Opening using another method, $Method_B$. Teamwork in this project is to find ways of putting the instances of enclosure and opening together so that instances of an emerging domain *Envelope* can be formed[3]. The domain of envelope is considered to be built up jointly by A and B on the bases of A's enclosure and B's opening. Remembering that this is a contrived example, we now describe the spatial concepts developed as $Method_A$ and $Method_B$ below.

5.2.1 The Enclosure modelled by Designer A

Designer A, as an expert in designing spatial enclosure, has the following working principle, described as $Method_A$, which is supposed to be used by A for the task of

[3]Think of, Enclosure as referring to walls, or other elements, with the purpose of keeping the weather out. Opening can be thought of as windows, being elements added to allow light in and a view of the outside. Envelope can be thought of as the combination of Enclosure and Opening, encompassing the purposes of both.

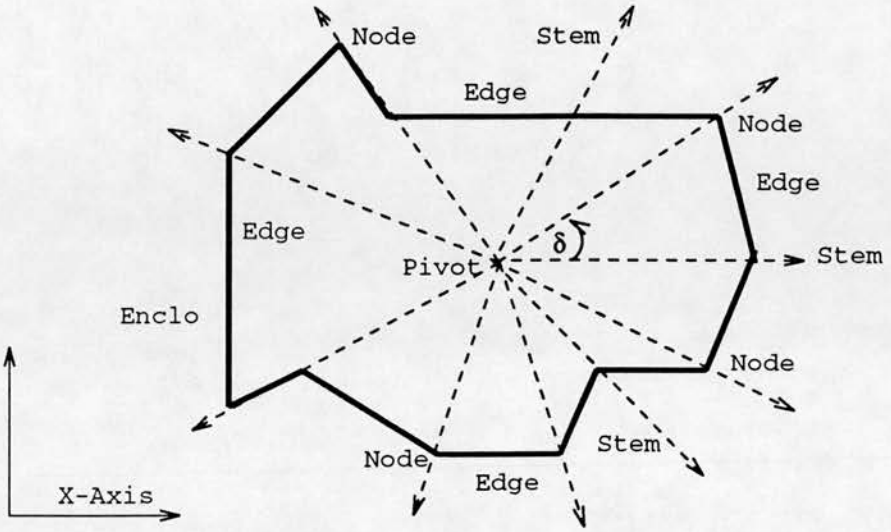


Figure 5.1. An instance of *Enclo* with an underlying conceptual schema specified as $Method_A$ by Designer A.

modelling the shapes of enclosure:

> $Method_A$ (see Figure 5.1): For an enclosure to exist, there is an initial point named as *Pivot*. A number of directed lines, *Stems*, pass through this point and are defined by the position of pivot and their angles. Up to two *Nodes* can be located on each stem according to their distances from the pivot. Any segment connecting two nodes defines an *Edge*. A closed connected chain of edges then forms a closed area of *Enclo*, an enclosure in a two-dimensional space. An enclosure thus formed has the attributes of length and area; the length of an enclosure is the total of all the lengths of its constituent edges, and the area of an enclosure is the total areas of its constituent triangles defined by its edges and the pivot.

5.2.2 The Opening modelled by Designer B

Suppose Designer B, being experienced in window design, has a particular conception of what is required for a good opening design in responding to a given building site. We assume that $Method_B$ is defined by B as follows:

> $Method_B$ (see Figure 5.2): In deciding openings for a building, we may start with a directed view line, named as *Vl*, which is decided by giving

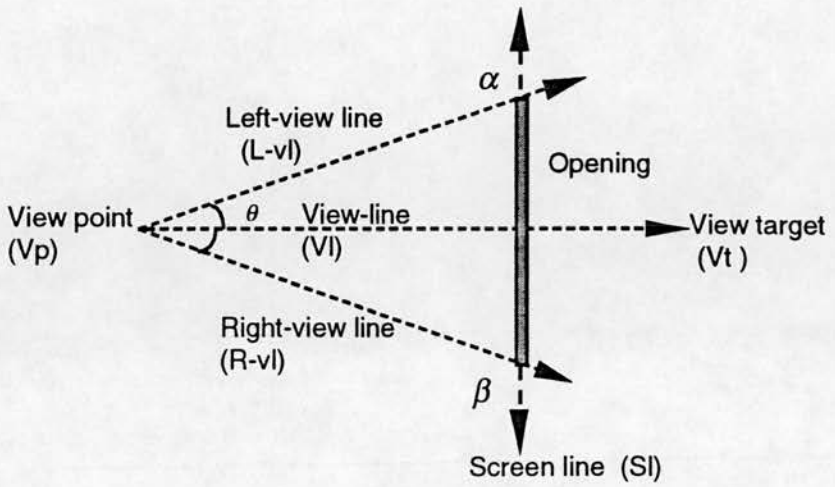


Figure 5.2. An instance of *Opening* with an underlying conceptual schema specified as $Method_B$ by Designer B.

two initial reference points, view point, *Vp*, and view target, *Vt*. Two directed lines, left view line, *L-vl*, and right view line, *R-vl*, are defined by the position of the view point and view angles. Perpendicular to the view line, a screen line, *Sl*, can be located by giving a distance to the view point, *Vp*. An *Opening* is then determined by the intersections of the left view line, and right view line with the screen line. We might thus name the two ends of an opening as α and β respectively; an opening has the attribute of length which is decided by the distance between its α and β ends.

5.2.3 The Envelope modelled jointly by A and B

As shown in the above, apart from some basic geometric elements such as *point*, *line*, *directed line*, and *distance between points* etc., the two methods for modelling instances of opening and enclosure are defined independently of each other. That is, it is not necessary for Designer A to know what Designer B is defining when he is defining $Method_A$, and vice versa.

Suppose in a design meeting, Designers A and B are motivated or invited to put their domains of expertise together such that a design of *Envelope* can be developed on the bases of opening and enclosure. Given the initiative of combining multiple source shapes of opening and enclosure into a resultant shape of *Envelope*, a common

task will be considered by A and B is how their individual modelling methods may be interrelated. More specifically, a common goal for A and B to achieve is to find new *abstractions* of greater interpretative power with which more complex spatial structures can be modelled by an integration of simpler ones originated from different design domains. To carry out the joint abstraction of integrative functions for modelling shapes of envelope, as we may think of this from the graphical examples presented above (for a more elaborate illustration, see Figure 5.5 on page 121), A and B have to jointly resolve the interpenetration between opening's disruption of the closeness of enclosure and enclosure's blocking the views of opening.

5.3 The Simulation Platform: OBJ3

The formal system used to implement a simulation of the above design exercise is the *modular algebraic specification* supported by the functional language OBJ3. A full account of the formal basis of OBJ3 can be found in the original authors' technical report of the language [GW88]. Being relevant to our exercise of shape construction, an application of modular algebraic specification to some basic geometrical constructions can be found in [Gog89]. For our current purpose, other formal systems, notably, VDM [Jon86] and Z [Spi89] are not used since they are not machine executable languages. Since OBJ3 is also used in another simulation for the structuralist requirements in Chapter 7, a brief introduction to the language is provided in this section. In our view, the best way to introduce the basic ideas and operations of OBJ3 is to go through a number of examples related to the theme of shape construction. Consider the following examples.

Example 1 (Point) *Consider in 2-dimensional space, a* **point** *is defined by its Cartesian coordinates* (a, b)*, in which* a *is the* **x** *coordinate,* b *the* **y** *coordinate. Let* A *and* B *be points in the plane with Cartesian coordinates* (x_1, y_1) *and* (x_2, y_2)*, it follows from Pythagoras' Theorem that the* **distance** $|AB|$ *is equal to* $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$*. The* **midpoint** *of* AB *has coordinates* $(1/2(x_1 + x_2), 1/2(y_1 + y_2))$*. How can Point be specified in OBJ3?*

```
obj POINT is
  protecting FLOAT .
  sort Point .
  op point : Float Float -> Point .
```

```
  ops x y : Point -> Float .
  op distance-pp : Point Point -> Float .
  op midpoint : Point Point -> Point .
  vars Float1 Float2 : Float .
  vars P1 P2 : Point .
  eq x(point(Float1,Float2)) = Float1 .
  eq y(point(Float1,Float2)) = Float2 .
  eq distance-pp(P1,P2) = sqrt((y(P2)-y(P1))*(y(P2)-y(P1))+
                               (x(P2)-x(P1))*(x(P2)-x(P1))) .
  eq midpoint(P1,P2) = point((x(P1)+x(P2))/2,(y(P1)+y(P2))/2) .
endo
```

In the above specification of the object *Point*, `POINT`, `obj` $\cdots$ `is`, `protecting`, `sort`, `op(s)`, `var(s)`, `(c)eq`, and `endo` are system-defined keywords; and `FLOAT` is a system built-in module (i.e., an OBJ3 implementation of floating or real number); `sqrt` is a system built-in function calculating the square root of a floating-point number. All other expressions such as object name `POINT`, sort name `Point`, predicate names are used at our discretion. More specifically, what we have presented in the above specification is basically to give the construct 'point' a mathematical model of an equational theory consisting of two parts:

- a set of symbols, Σ, denoting values of the sorts (types). This set is referred to as the *carrier set* of the algebra;
- a set of closed operations, E, on the carrier set, possibly including nullary, constant, and operations. Properties (semantics) of operations are defined by equations.

The algebra that models our *Point* with the operations `point` (construct a point out of two floating numbers), `x` and `y` (select the x and y coordinates of a point), `distance-pp` (evaluate the distance between two points), `right-to` (evaluate if a point is to the right of another point), and `midpoint` (construct a point which is the middle point of two given points) is

$$[(BOOL, FLOAT, POINT), point, x, y, distance - pp, right - to, midpoint]^4$$

In Abstract Data Type terms, the form of specifying the syntax of an operation is called *signature*. Therefore, in our equational theory presentation of "point", we have

[4]The sort `BOOL` has been previously imported into `FLOAT` by the OBJ3 system. Since we import `FLOAT` into `POINT`, all the sorts, operations, and equations defined into `FLOAT` and `BOOL` are also imported into `POINT`. For simplicity, they are not displayed here.

the following signature for 'point':

source data		*operations*		*resultant data*
`Float Float`	`---->`	`point`	`---->`	`Point`
`Point`	`---->`	`x`	`---->`	`Float`
`Point`	`---->`	`y`	`---->`	`Float`
`Point Point`	`---->`	`distance-pp`	`---->`	`Float`
`Point Point`	`---->`	`right-to`	`---->`	`Bool`
`Point Point`	`---->`	`midpoint`	`---->`	`Point`

We may continuously add new signatures into the current version of `POINT`. But we are now in a position to *reduce* syntactically valid expressions in OBJ3. When `POINT` is input to OBJ3, the environment, including the built-in modules, `BOOL`, `FLOAT`, and `POINT`, can compute the results of given expressions to normal form by the system's *term-rewriting* mechanism. In `POINT`, for instance, we can construct a point with a pair of real numbers $(12, -12)$, or evaluate the distance between $point(2, 3)$ and $point(-9, 25)$, or select the x coordinate of the middle point constructed by $point(2, 6)$ and $point(-3, 8)$:

```
OBJ> reduce point(12,-12) .
reduce in POINT : point(12.0,-12.0)
rewrites: 0
result Point: point(12.0,-12.0)

OBJ> reduce distance-pp(point(2,3),point(-9,25)) .
reduce in POINT : distance-pp(point(2.0,3.0),point(-9.0,25.0))
rewrites: 17
result Float: 24.597

OBJ> reduce x(midpoint(point(2,6),point(-3,8))) .
reduce in POINT : x(midpoint(point(2.0,6.0),point(-3.0,8.0)))
rewrites: 10
result Float: -0.5
```

Example 2 (Angle) *In a Cartesian coordinate system, we define the* **angle** *of a 2-dimensional vector as a* **degree** *or a* **radian** *value. For an angle in radians, it is represented by a real number R. Given an angle D in degrees, the angle is converted to R in radians by $R = \pi * (D/180)$, where $0 \leq R \leq 2\pi$ if $0 \leq D \leq 360$. For an*

angle in degrees, it is constructed by a given quadruple of real numbers (x_1, y_1, x_2, y_2). *Let* $X = (x_2 - x_1)$ *and* $Y = (y_2 - y_1)$. *If* (X, Y) *is in the first quadrant, the angle of* $(x_1, y_1, x_2, y2)$ *is equal to* $tan^{-1}((y_2 - y_1)/(x_2 - x_1))$ *in radians. If* (X, Y) *is in the second quadrant, the angle of* (x_1, y_1, x_2, y_2) *is equal to* $(\pi - tan^{-1}(|y_2 - y_1|/|x_2 - x_1|)$ *in radians. If* (X, Y) *is in the third quadrant, the angle of* $(x_1, y_1, x_2, y2)$ *is equal to* $(\pi + tan^{-1}(|y_2 - y_1/|x_2 - x_1))$ *in radians. If* (X, Y) *is in the fourth quadrant, the angle of* $(x_1, y_1, x_2, y2)$ *is equal to* $(2\pi - tan^{-1}(|y_2 - y_1|/|x_2 - x_1|)$ *in radians. The angle of* $(x_1, y_1, x_2, y2)$ *is equal to 0 in degrees, if* $y_2 - y_1 = 0$ *and* $x_2 - x_1 > 0$. *The angle of* $(x_1, y_1, x_2, y2)$ *is equal to 90 in degrees, if* $y_2 - y_1 > 0$ *and* $x_2 - x_1 = 0$. *The angle of* $(x_1, y_1, x_2, y2)$ *is equal to 180 in degrees, if* $y_2 - y_1 = 0$ *and* $x_2 - x_1 < 0$. *The angle of* $(x_1, y_1, x_2, y2)$ *is equal to 270 in degrees, if* $y_2 - y_1 < 0$ *and* $x_2 - x_1 = 0$. *An OBJ3 specification of Angle is shown below.*

```
obj ANGLE is
   protecting FLOAT .
   sorts Radian Degree .
   subsorts Float < Radian .  *** Float is a subsort of Radian
   subsorts Float < Degree .  *** Float is a subsort of Degree
   op inradian : Float -> Radian .
   op indegree : Float -> Degree .
   op angle : Float Float Float Float -> Degree .
   op d-to-r : Degree -> Radian .
   op r-to-d : Radian -> Degree .
   vars X1 Y1 X2 Y2 : Float .
   var D : Degree .
   var R : Radian .
   cq angle(X1,Y1,X2,Y2) = r-to-d(atan((Y2-Y1)/(X2-X1)))
                        if Y2-Y1 > 0   and   X2-X1 > 0 .
   cq angle(X1,Y1,X2,Y2) = 180-r-to-d(atan(abs(Y2-Y1)/abs(X2
       -X1)))           if Y2-Y1 > 0   and   X2-X1 < 0 .
   cq angle(X1,Y1,X2,Y2) = 180+r-to-d(atan(abs(Y2-Y1)/abs(X2
       -X1)))           if Y2-Y1 < 0   and   X2-X1 < 0 .
   cq angle(X1,Y1,X2,Y2) = 360-r-to-d(atan(abs(Y2-Y1)/abs(X2
       -X1)))           if Y2-Y1 < 0   and   X2-X1 > 0 .
   cq angle(X1,Y1,X2,Y2) = 0    if Y2-Y1 == 0 and X2-X1 > 0 .
   cq angle(X1,Y1,X2,Y2) = 90   if Y2-Y1 > 0 and X2-X1 == 0 .
   cq angle(X1,Y1,X2,Y2) = 180  if Y2-Y1 == 0 and X2-X1 < 0 .
   cq angle(X1,Y1,X2,Y2) = 270  if Y2-Y1 < 0 and X2-X1 == 0 .
   eq d-to-r(D) = pi*(D/180) .
   eq r-to-d(R) = R*(180/pi) .
endo
```

In the above example, we have shown how to declare the *subsort(s)* relations among sorts and how to give operational semantics to predicates by *conditional equations* in OBJ3. With **ANGLE** we can now reduce the following expressions:

```
OBJ> red angle(1,2,3,4) .
reduce in ANGLE :
angle(1.0,2.0,3.0,4.0)
rewrites: 14
result Float: 45.0

OBJ> red angle(12,-8,12,8) .
reduce in ANGLE : angle(12.0,-8.0,12.0,8.0)
rewrites: 31
result Float: 90.0
```

Since we have specified `Point` and **Angle** respectively, how can we combine point with angle expressions in the construction of an angle? For instance, can we construct an angle out of the values of two given points?

```
?? reduce angle(x(point(7.2,-3.5)),y(point(7.2,-3.5)),x(point(-8.56,
-11.23)),y(point(-8.56,-11.23))) .
```

Because of the lack of *interconnectivity* between **POINT** and **ANGLE**, the current OBJ3 database does not recognize the above expression (that is, the current **ANGLE** does not have a specification of the **POINT** data type; the **angle** function cannot be applied with `point` as its arguments.). The issue of interconnectivity is an important notion in modular algebraic specification. We use the next example *Line* to show a combination of **POINT** and **ANGLE** in **LINE**.

Example 3 (Line) *In line, we consider (a) construct a line of two given different points, point*(x_1, y_1)*, and point*(x_2, y_2)*, where* $x_1 \neq x_2$ *or* $y_1 \neq y_2$*. (b) construct an intersection of two given different lines,* $L_1(point(f_1, f_2), point(f_1', f_2'))$ *and* $L_2(point(g_1, g_2), point(g_1', g_2'))$*, where one of the four given points is different from the other three points. (c) evaluate if a given point,* $point(x, y)$ *is on a given line,* $line(point(x_1, y_1), point(x_2, y_2))$*. For (a), given the importation of* **POINT***, a line L is constructed under two conditions:*

(a.1) $line(point(x_1, y_1), point(x_2, y_2))$*, where* $x_1 \neq x_2$ *or* $y_1 \neq y_2$*.*

(a.2) $line(point(x_1, y_1), point(x_2, y_2))$ *is a* **vertical line** *if* $x_1 = x_2$*.*

For (b), given the importation of `POINT` *and* `ANGLE`*, the intersection point*(X, Y) *of two given lines is constructed as follows under different circumstances:*

(b.1) *If* L_1 *is parallel to* L_2*, there is no intersection.*

(b.2) *If* L_1 *is not parallel to* L_2*, and both* L_1 *and* L_2 *are general, i.e., not vertical, the intersection is calculated by the equations*

$$X = \frac{b_2 - b_1}{a_1 - a_2}, \quad Y = \frac{a_1 b_2 - a_2 b_1}{a_1 - a_2}; \quad \textit{where}$$

$$a_1 = \frac{f_2 - f_2'}{f_1 - f_1'}, \quad b_1 = \frac{f_2' f_1 - f_2 f_1'}{f_1 - f_1'}, \quad a_2 = \frac{g_2 - g_2'}{g_1 - g_1'}, \quad b_2 = \frac{g_2' g_1 - g_2 g_1'}{g_1 - g_1'}$$

(b.2.1) L_1 *is not parallel to* L_2*, if* $angle(f_1, f_2, f_1', f_2') - angle(g_1, g_2, g_1', g_2') \neq 0, 180, -180$.

(b.2.2) *Both* L_1 *and* L_2 *are general, if* $f_1' \neq f_1$ *and* $g_1' \neq g_1$.

(b.3.1) *If* L_1 *is vertical, and* L_2 *is general, then the intersection point*(X, Y) *is given by*

$$X = f_1, \quad Y = \frac{g_2(g_1' - f_1) + g_2'(f_1 - g_1)}{g_1' - g_1}$$

(b.3.2) *If* L_1 *is general, and* L_2 *is vertical, then the intersection point*(X, Y) *is given by*

$$X = g_1, \quad Y = \frac{f_2(f_1' - g_1) + f_2'(g_1 - f_1)}{f_1' - f_1}$$

For (c), given the importation of `POINT`*, whether a given point is on a given line is evaluated in two conditions:*

(c.1) *If* $line(point(x_1, y_1), point(x_2, y_2))$ *is not vertical, and* $(y - y_1)/(x - x_1) = (y_2 - y)/(x_2 - x)$*, then the point is on the line.*

(c.2) *If* $line(point(x_1, y_1), point(x_2, y_2))$ *is vertical, and* $x = x_1 = x_2$*, then the point is on the vertical line.*

A full specification of `LINE` in OBJ3 is given in Appendix B.3 of the thesis. Our previous combined expression in `POINT` and `ANGLE` is now in a valid form for processing in `LINE`:

```
OBJ> reduce angle(x(point(7.2,-3.5)),y(point(7.2,-3.5)),
```

```
                x(point(-8.56,-11.23)),y(point(-8.56,-11.23))) .
reduce in LINE : angle(x(point(7.2,-3.5)),y(point(7.2,-3.5)),x(point(
                -8.56,-11.23)),y(point(-8.56,-11.23)))
rewrites: 46
result Float: 206.127
```

Apart from the above POINT, ANGLE, and LINE examples, following the same approach, we can also specify LINE-PA (for constructing a line by a point and an angle), SEGMENT, ARC, CIRCLE, etc., as a collection of basic (unchanging) geometric entities in OBJ3. Suppose the collection of well-defined system entities is given at the outset, then we shall have a formal basis for collaborative spatial constructions participated by designers using different modelling methods.

5.4 From Enclosure and Opening to Envelope

We have given some examples of modular algebraic specification in the language OBJ3. As shown, it is with the mathematics of *order-sorted algebra* that we can achieve structural representations of simple geometrical constructs as 'mind-sized' modules; the separation of signature and predicates in a module facilitates *typed data abstraction*; and the interconnectivity among distributed modules can be neatly obtained on the basis of *modularisation*. Using the computational mechanisms, it is straightforward to produce symbolic representations of the *Enclosure* and *Opening* methods described informally earlier. In a bottom-up and stepwise manner, for each modelling method, the constituent constructs are simulated in OBJ3 modules (e.g., node, stem, edge etc. in enclosure). By putting the modules specified into hierarchical structures, the complete modelling methods can be symbolically represented. Instances of enclosure and opening can then be produced on the OBJ3 platform. Full specifications of $Method_A$ (enclosure) and $Method_B$ (opening) are presented in Appendix C.1 and Appendix C.2 respectively.

5.4.1 A scheme of spatial operations

When the participating methods are made available, we are in a position to simulate how a shape of enclosure might be integrated with a shape of opening. First of all, this involves new constructs and operations to emerge from the two existing methods. To simulate the joint abstraction of integrative functions, we have tried out a descriptive scheme called *spatial operations*. Basically, a single spatial operation can be considered as a new function that emerges from an interconnection of constituent concepts specified in the individual modelling methods. Interconnection is mainly motivated by

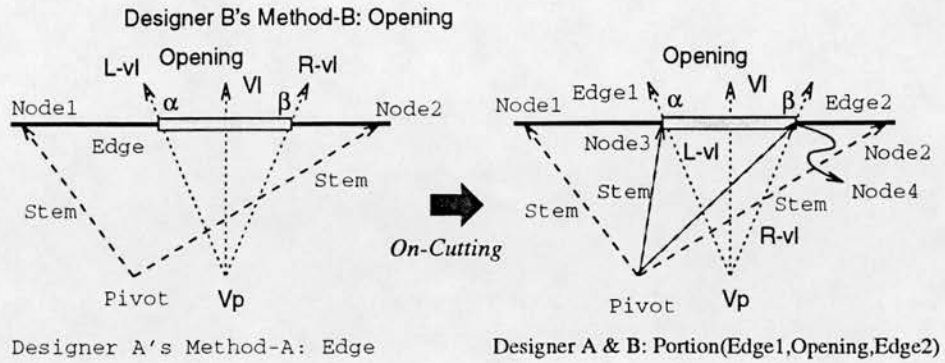


Figure 5.3. The *On-Cutting* operation can be applied to a pair of edge and opening, if the α and β ends of an opening are collinear with the nodes of an edge. The result, as intended jointly by A and B, is a 'portion' configuration of two emerging edges and the given opening.

considering how instances of shapes from different domains might be spatially correlated with one another. To illustrate the scheme, two example spatial operations are described informally below.

1. The *On-Cutting* operation is devised to transform, more specifically, to cut an edge (of $Method_A$) with an imposed opening (of $Method_B$) such that a new element, named Portion, can be generated with a structure of two 'emerging' edges and the given opening. To see the structure in detail, a description of *On-Cutting* together with its depiction is given as follows.

   > **Spatial Operation [On-Cutting]** (see Figure 5.3): Given an Edge and an Opening, if the α end and the β end are *on* the Edge, then a Portion is constructed by (1) cutting the Edge with the Opening and yielding two Edges, (2) combining the two Edges of (1), with the Opening, into a Portion.

   A full specification of the spatial operation *On-Cutting* is provided in Appendix C.3. Two testing cases of applying *On-Cutting* onto instances of enclosure and opening are presented in Appendix C.5.

2. The second example is considered in dealing with a different spatial correlation between an edge and an opening. For some (e.g. getting a better view or more light), an opening has to be displaced 'outside' an enclosure. For this, A and

B are motivated to define another way of breaking an edge and connecting the resultant edges with the given opening. Again, the *Out-Cutting* operation cannot be fully defined without accessing the structures and instances of both edge and opening. The following description and diagram show how the *Out-Cutting* operation might be specified.

> **Spatial Operation [Out-Cutting]** (see Figure 5.4): Given an edge and an opening, if the opening is *outside* the enclosure, then a 'portion' is defined by connecting the opening with the edge in the following steps:
>
> - Draw a line l_1 parallel to Vl (the View-line of Opening) through either end of the opening if it is parallel to the edge, or through the end of the opening that is *farther from* the edge, and get the intersection i_1 of l_1 and the edge;
> - Draw a line l_2 perpendicular to the edge through the other end of the opening if it is parallel to the edge, or through the end of opening that is *nearer to* the edge, and get the intersection i_2 of l_2 and the edge;
> - From left to right, make the following four edges: Edge1(Node1, i_1) Edge2(i_1, α), Edge3(β,i_2), Edge4(i_2, Node2);
> - Portion is then defined by: Portion(Edge1, Edge2, Opening, Edge3, Edge4).

A full specification of the spatial operation *Out-Cutting* is presented in Appendix C.4.

5.4.2 The resultant enclosure and opening

To compute the correlation such as an opening *across*, or *inside* an edge, many other spatial operations might be developed, but to keep the illustration within a reasonable length they are not presented here. The point is that the repository of spatial operations can always be extended whenever A and B identify new ways of correlating the shapes modelled in each domain. Clearly, in comparison with $Method_A$ and $Method_B$, the collection of spatial operations can be applied to 'combined' instances of enclosure and opening jointly given by A and B as 'source shapes'. Depending on what spatial relations are held between the source shapes (e.g., an edge is 'on', or, 'out' an opening), specific spatial operations are applied, resulting in a *composite shape* of envelope.

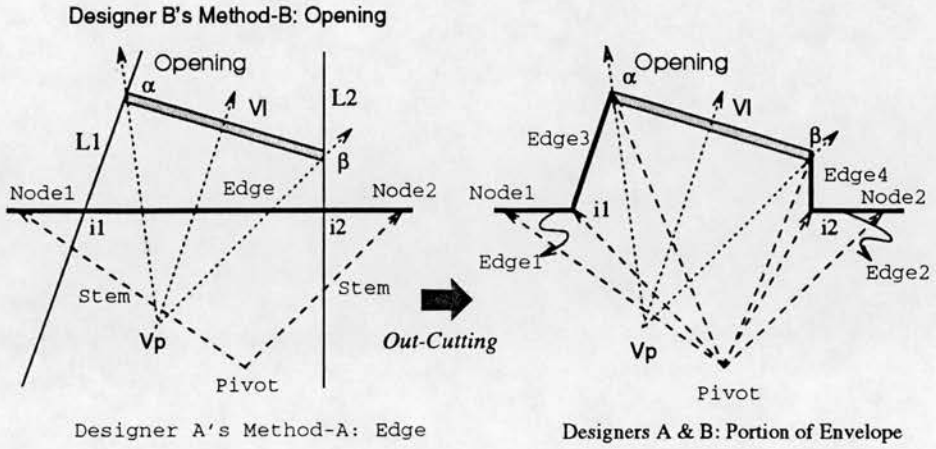


Figure 5.4. The *Out-Cutting* operation can be applied to a pair of edge and opening, if the opening is placed outside the edge regarding the pivot. The result is a 'portion' configuration of four emerging edges and the given opening.

Obviously, different spatial operations will result in different integration of enclosure and opening into envelope.

Our OBJ 3 simulation also shows that, in an envelope, the *resultant shapes* of enclosure and opening remain distributable to A's and B's domains if the retrieval functions (i.e., the operations to extract an enclosure or a set of openings from a given envelope) are specified. Figure 5.5 depicts an example of an integration of source shapes and a distribution of the resultant shapes. This is a case of the use of specified spatial operations, and it illustrates that the use of spatial operations must be a joint effort:

- Before integration, Designer A and B need to prepare jointly a combined source shape, which consists of original shapes constructed in their individual methods.
- After integration, Designer A and B need to communicate with each other for the resultant envelope, as, for example, Designer A may see some problems after A's evaluation of the resultant enclosure. In a sense, each participant sees the design consequence of his or her domain design proposal in the light of the integrated envelope image.

What has been demonstrated in the above is a symbolic simulation of joint abstraction of $\mathcal{SIS}$ in terms of the formal specification of spatial operations. We may say that a collection of spatial operations constitute an integration schema shared by

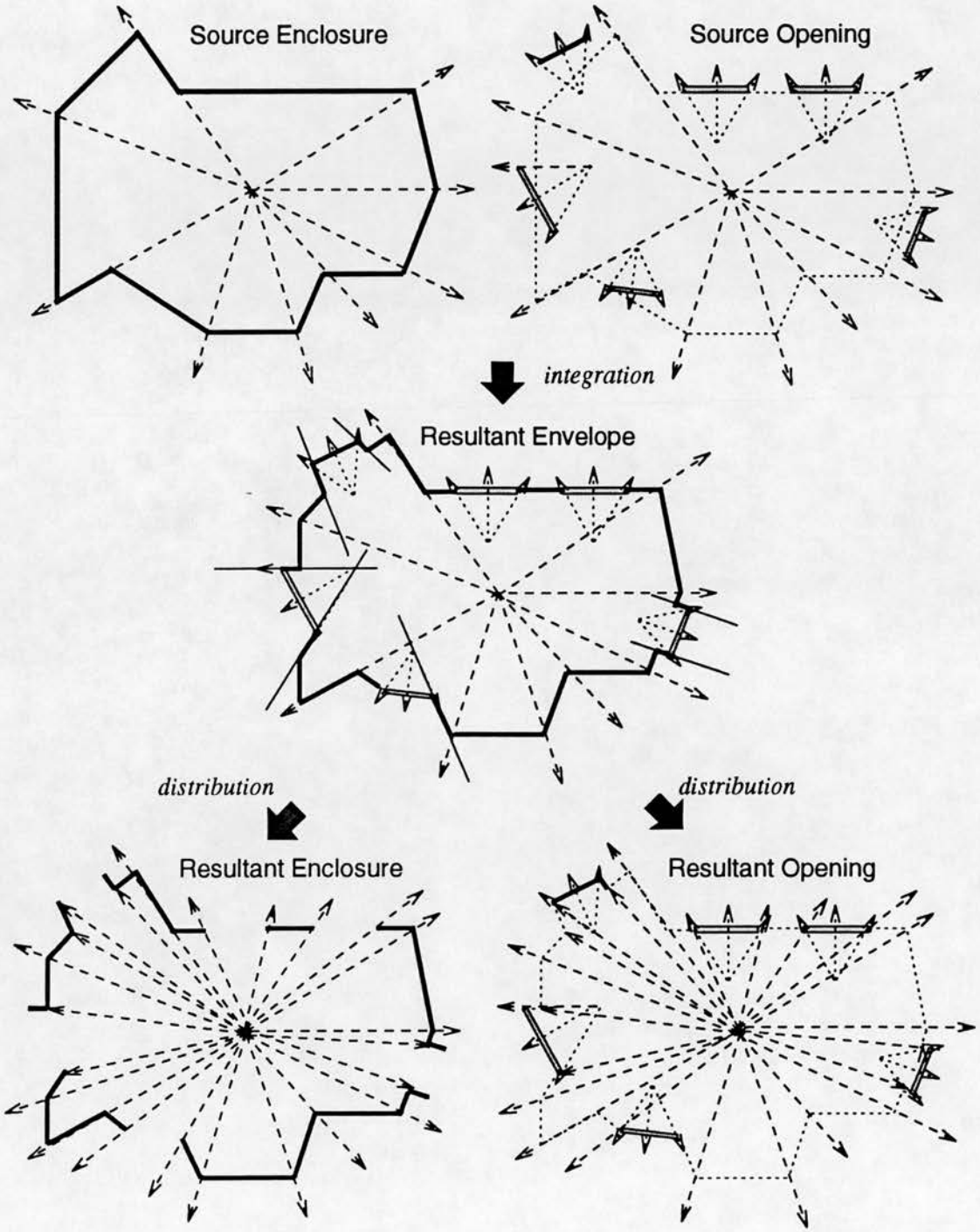


Figure 5.5. A case of integrating a source enclosure and several source openings into a resultant envelope from which the resultant enclosure and openings can be retrieved.

their designers. Clearly, any specification of a spatial operation must be a collaborative effort that requires joint contributions from each of the participating areas of expertise. Our OBJ3 simulation also shows that some form of collaboration among participants is necessarily involved in any use of spatial operations to arrive at integrated designs as common images. To clarify where collaborations arise, we now turn to the internal structures of the collaboration tasks discernible in our simulation of joint abstraction and use of $\mathcal{SIS}$.

5.4.3 Co-specifying spatial operations

The specification of spatial operations is an essential part of group design, without which common images cannot be produced. Basically, it involves a group process of correlating the spatial constructs introduced by one participant with those by others. When designers have jointly specified a number of spatial operations, we say that they reach a state of sharing an integration schema. Given the structure of a spatial operation as simulated in the algebraic specification system, joint specification of a spatial operation involves the following:

- *name* — the names like *On-Cutting, Out-Cutting* in our examples. How spatial operations get named is dependent on the agreements among the authors so that they can be commonly recallable to all parties later on;
- *denotations* — declarations of integrative functions made of domain constructs specified in individual modelling methods. No one designer has complete and precise knowledge about what modules to import in declaring the signature of an operation;
- *operations* — formulations of operational semantics following the signatures denoted. Operations are formulated to do certain things such as *translation, selection, reconstruction* etc. The formulation of operational semantics for a declared function requires participants to coordinate in giving proper translations and inclusions of domain constructs to match the denotation of a targeted construct;
- *conditions* — formulations of operational semantics which computes if certain spatial relations are held between instances of source shapes (for instance, to compute if an opening is *located on* an edge). The formulation of conditions requires participants to coordinate in giving proper translations of domain constructs to fit with the denotation of a targeted function that checks a geometric relation (e.g., a point is on a line).

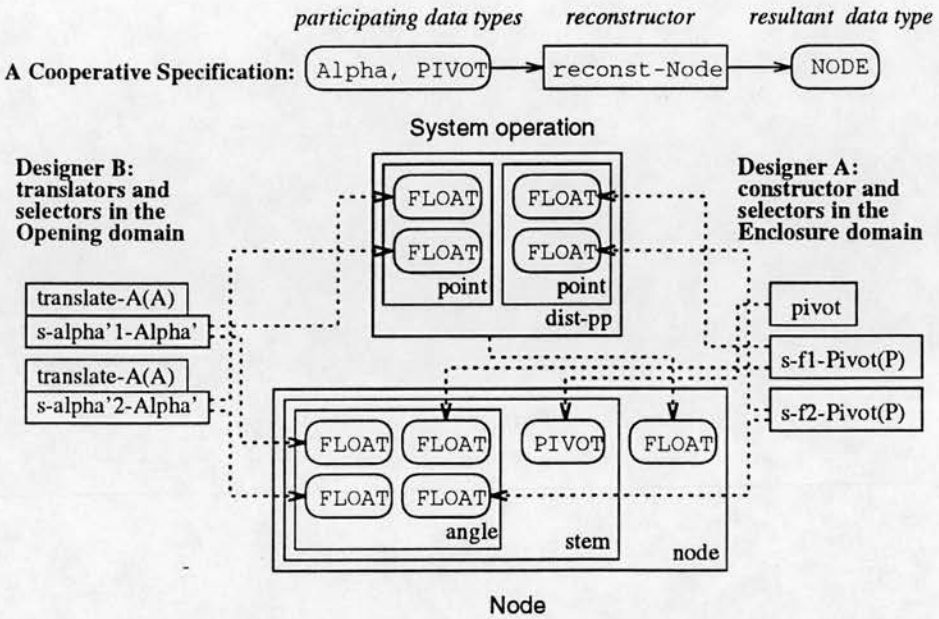


Figure 5.6. A structure diagram shows the cooperation for a joint specification of reconstructing `Node` with `Alpha` and `Pivot`.

The structure diagram in Figure 5.6 shows what might be involved if the operation of reconstructing `Node` with `Alpha` and `Pivot` is co-specified by Designers A and B[5]. This *reconstructor* is one of the several functions contributing to a full specification of the spatial operation *On-Cutting* (see Appendix C.5). The data type `Node` (of $Method_A$) is reconstructed (as in contrast to its original construction) with the source data types of `Alpha` (of $Method_B$) and the `Pivot` (of $Method_A$). In specifying the reconstructor `reconst-Node`, Designer A, the person has the knowledge of `Node`, is able to achieve this specification if and only if Designer B collaborates with A by providing his knowledge of translating `Alpha` to `Point` and to two `Floats`. With these translations available, a shape of Node can be generated in a pair of the α end of an opening and the pivot of an enclosure.

The above diagram depicts a formal structure of cooperative specification. Firstly, there is a *common target* expressed by "`Alpha` × `Pivot` → `Node`"[6]. As shown in the diagram, for A and B to achieve a joint specification, what matters is the algebraic

[5]The term *structure diagram* is borrowed from Bergstra *et al.* [BHK89]. The words in `verbatim` style are now used to denote that they are formally specified as abstract data types in OBJ3.

[6]This reconstructor `reconst-Node` is an overloaded function; it has another targeted form of "`Beta` × `Pivot` → `Node`".

structures of `Node` and that of the system operation `dist-pp`, which computes the distance between two given points. Seen from this example, the lowest common ground for joint abstraction seems to lie in a set of primitive constructs that domain constructs can be translated into. We shall discuss what this implies for system strategies later on.

5.4.4 Joint provision of source shapes

The use of spatial operations for generating integrated shapes as potential common images presents another focal task of cooperation. The need for teamwork arises because successful uses of any spatial operations depend on putting together the source shapes of each participating domain in a way that matches the algebraic structures specified in the operations. However, the provision of combined source shapes need not be as restrictive as this. Participants can explore possible 'dependent' relations among parts of source shapes.

As an example, Figure 5.7 shows an OBJ3 simulation of a joint provision of combined source shapes, in which the opening shape is constructed as being dependent on the edge shape, and a composite shape of portion is computed by the spatial operation *On-Cutting*. As shown, the two source shapes are combined in a way that the `Opening` is always to be *on* the `Edge`. For Designer B to achieve this dependent construction, a `Segment` representation of `Edge` has to be made available by Designer A; what A has to do in this case is to provide the mapping from `Edge` to `Segment` which is within A's knowledge domain. Without incorporating the mapping, B is unable to fulfil his intention of modelling the dependent relation among the source shapes.

Drawing upon from this and other examples we have tried out, it can be remarked that all the system primitives can be potentially employed for mapping parts of a source shape in one domain into a primitive format, which can then be included in other source shapes. In consequence, what can be obtained from a joint provision of source shapes is the generation of a composite shape that contains resultant sub-shapes distributable to the individual modelling spaces.

5.4.5 Coordination in making design changes

The third aspect of cooperation lies in the coordination among designers in modifying domain source shapes. A natural consequence of dependent constructions in joint provisions of source shapes, as described above, is the propagation of design changes to related source and resultant shapes. Why coordination is necessarily involved may

OBJ3 EXPRESSIONS **EXPLANATIONS**

```
reduce mkPortion
(edge(node(stem(pivot(20,7),150),20),
      node(stem(pivot(20,7),30),20)),

isOpen(isAlpha(sl(vl(vp(28,15),vt(28,70)),
 distance-pp(point(28,15),intersect(
 segment(point(28,15),point(28,70)),
 segment(node-to-point(node(stem(pivot(
          20,7),150),20)),
          node-to-point(node(stem(pivot(
          20,7),30),20))))),
 l-vl(vp(28,15),150)),
      isBeta(sl(vl(vp(28,15),vt(28,70)),
 distance-pp(point(28,15),intersect(
 segment(point(28,15),vt(28,70)),
 segment(node-to-point(node(stem(pivot(
          20,7),150),20)),
          node-to-point(node(stem(pivot(
          20,7),30),20))))),
 r-vl(vp(28,15),30)))) .
```

A's expression of an Edge shape

B's expression of an Opening shape which is constructed dependently on the A's expression by including the mapping between the Segement and Edge provided by A (shown in the underlined parts of expression)

A composite shape of Portion has the following parts:

```
rewrites: 3123
result Portion:on-cut(
edge(node(stem(pivot(20.0,7.0),150.0,20.0),
     node(stem(pivot(20.0,7.0),65.60),10.98)),

mkOpen'(mkAlpha'(24.54,17.0),mkBeta'(31.46,17.0)),

edge(node(stem(pivot(20.0,7.0),41.10),15.21),
     node9stem(pivot(20.0,7.0),30.0),20.0)))
```

Resultant sub-shape1 of Edge

Resultant sub-shape2 of Opeing'

Resultant sub-shape3 of Edge

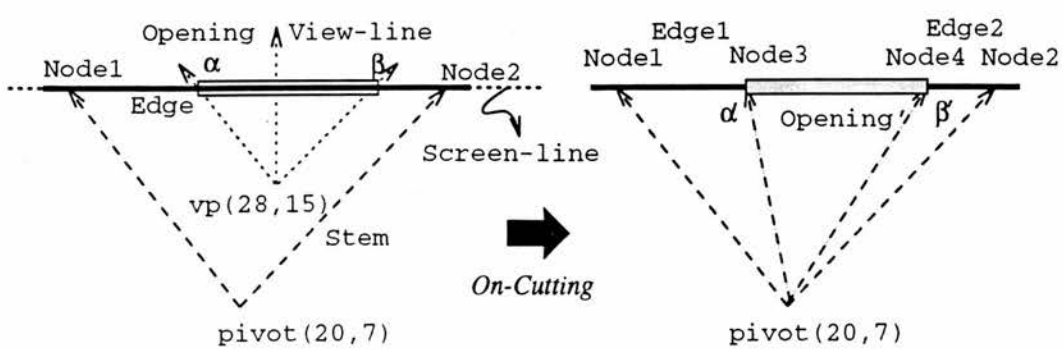


Figure 5.7. An OBJ3 simulation of a joint provision of combined source shape where the opening shape is made jointly by A and B to be *on* the edge shape.

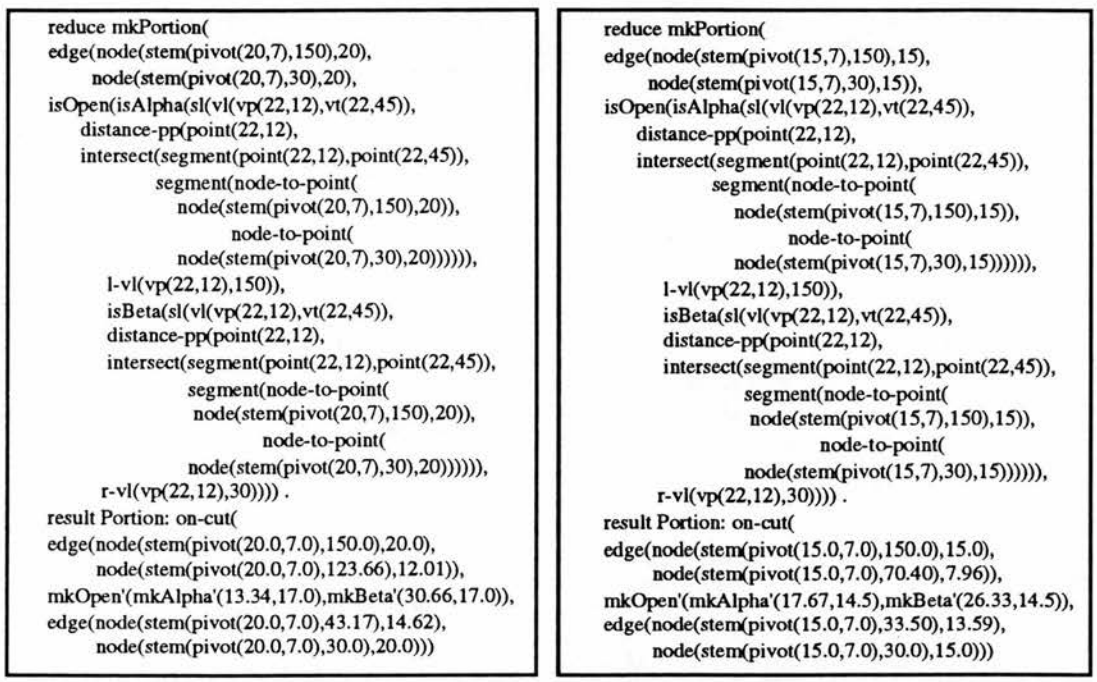
be better explained by the following account:

> Assume two designers, (say), A and B, are engaged in a joint provision of source shapes. We denote that the source shape with the sort Ω modelled by A as SR_Ω, and the one with the sort Ψ by B as SR_Ψ. A joint source shape can be provided as $j(SR_\Omega, SR_\Psi)$ if the j operation is selected for a valid shape combination jointly intended by A and B. By executing $j(SR_\Omega, SR_\Psi)$, a composite shape with the sort Φ, denoted as CR_Φ, can be generated in the shared modelling space.
>
> In the case that SR_Ψ is constructed dependently on SR_Ω by B, then changes made in SR_Ω (resulting ΔSR_Ω) by A at some later moment will consequently alter the current state of SR_Ψ and that of CR_Φ. That is, the alteration will be propagated to the constituents of CR_Φ, i.e., the resultant sub-shapes of RSR_Ω and $RSR_{\Psi'}$, as in $CR_\Phi = k(RSR_\Omega, RSR_{\Psi'})$ where k is a label indicating what particular spatial operation has been applied. Note that RSR_Ω and $RSR_{\Psi'}$ are recognisable and distributable to A's and B's modelling spaces respectively. Therefore, the change of SR_Ω to ΔSR_Ω made by A shall generate the changing composite shape, $\Delta CR_\Phi = k(\Delta RSR_\Omega, \Delta RSR_{\Psi'})$, for which B may or may not agree with the impending $\Delta RSR_{\Psi'}$.

A juxtaposition of two examples in Figure 5.8 shows an OBJ3 simulation of making design changes. Overlapping a shape of `Edge` initially given by Designer A, Designers A and B jointly constructs an `Opening` shape (such that the latter is spatially located on the former). Given the combined source shapes to the shared integration schema, the *On-Cutting* is applied and generates a composite shape of `Portion` containing of two resultant sub-shapes of `Edge` and of `Opening'`[7]. Later on, A has a different consideration of the resultant edge and makes some changes (i.e., the pivot is moved to a new location with two shortened edges).

As a consequence of A's making changes in edges, alternations in the source shape of opening is followed due to the dependency constructed previously. When the on-cutting operation is applied, a new state of the portion shape will be computed, containing new states of resultant sub-shapes. Assuming that Designer B is able to receive the

[7]The `Opening'` data type is built upon `Alpha'` and `Beta'`, which are developed to make an opening shape in a way different from that of `Opening`, `Alpha`, and `Beta`, a case of multiple representations of the same design entity.



```
reduce mkPortion(
edge(node(stem(pivot(20,7),150),20),
     node(stem(pivot(20,7),30),20),
isOpen(isAlpha(sl(vl(vp(22,12),vt(22,45)),
    distance-pp(point(22,12),
    intersect(segment(point(22,12),point(22,45)),
            segment(node-to-point(
              node(stem(pivot(20,7),150),20)),
                  node-to-point(
              node(stem(pivot(20,7),30),20)))))),
      l-vl(vp(22,12),150)),
      isBeta(sl(vl(vp(22,12),vt(22,45)),
      distance-pp(point(22,12),
      intersect(segment(point(22,12),point(22,45)),
                segment(node-to-point(
                 node(stem(pivot(20,7),150),20)),
                       node-to-point(
                 node(stem(pivot(20,7),30),20)))))),
      r-vl(vp(22,12),30)))) .
result Portion: on-cut(
edge(node(stem(pivot(20.0,7.0),150.0),20.0),
     node(stem(pivot(20.0,7.0),123.66),12.01)),
mkOpen'(mkAlpha'(13.34,17.0),mkBeta'(30.66,17.0)),
edge(node(stem(pivot(20.0,7.0),43.17),14.62),
     node(stem(pivot(20.0,7.0),30.0),20.0)))
```

An integrated shape of Portion results from an application of the mkPortion operation.

```
reduce mkPortion(
edge(node(stem(pivot(15,7),150),15),
     node(stem(pivot(15,7),30),15)),
isOpen(isAlpha(sl(vl(vp(22,12),vt(22,45)),
    distance-pp(point(22,12),
    intersect(segment(point(22,12),point(22,45)),
            segment(node-to-point(
              node(stem(pivot(15,7),150),15)),
                  node-to-point(
              node(stem(pivot(15,7),30),15)))))),
      l-vl(vp(22,12),150)),
      isBeta(sl(vl(vp(22,12),vt(22,45)),
      distance-pp(point(22,12),
      intersect(segment(point(22,12),point(22,45)),
                segment(node-to-point(
                 node(stem(pivot(15,7),150),15)),
                       node-to-point(
                 node(stem(pivot(15,7),30),15)))))),
      r-vl(vp(22,12),30)))) .
result Portion: on-cut(
edge(node(stem(pivot(15.0,7.0),150.0),15.0),
     node(stem(pivot(15.0,7.0),70.40),7.96)),
mkOpen'(mkAlpha'(17.67,14.5),mkBeta'(26.33,14.5)),
edge(node(stem(pivot(15.0,7.0),33.50),13.59),
     node(stem(pivot(15.0,7.0),30.0),15.0)))
```

A change made in the Edge shape results in a change to the source Opening shape, and then generates a new Portion shape in which the resultant sub-shape of Opening has been displaced and reduced in length.

Figure 5.8. An OBJ3 simulation of making design changes in a combined source shape where dependent relations will give rise to communications and negotiations.

changing state of the resultant opening shape, which is his design responsibility, then B's perceiving the change might trigger further communications and negotiations with A. It can be the case that the change propagated to the resultant opening shape may be unacceptable to Designer B; or, B is able to suggest to A that further modifications in the opening source shape can result in an integrated envelope that appears more satisfactory than the current one.

5.5 Discussion

Believing that joint abstraction of shared integration schemas is an essential component of the metaphorist approach to collaborative design, we set out in this chapter a symbolic simulation. The simulation aims to clarify what cooperation tasks might be involved in the joint abstraction. We are interested in exploring the internal structures of collaboration where group members develop shared concepts and operations so that domain design contributions can be integrated into common images. A simple design exercise is set up indicating what is to be simulated. The exercise is a 'contrived' case for which we do not claim any 'utility' of the modelling methods exploitable in real world practice. In simulation, we also have deliberately constructed symbolic representations at a very low level, with which the imagined designers' actions are being analysed. Collaborative design in this simulation is described and depicted at some level of abstraction from whatever the actions actually are.

Like any other attempt to simulate complex processes, the above study has been preoccupied with some system-bound technical details that might distract us from the real purpose of carrying out the simulation. The following points are made for clarification and to serve as a note of the limitation of the present OBJ3 exercise:

- Given that we are dealing with two imaginary designers (and we are imagining them) each with their own imaginary methods—and we are imagining how the methods might be combined (by Designer A and B). The point of these illustrations is that any act of combining is motivated by participants' joint intention to put parts together into greater wholes and a support system must provide support for the participants' doing this. In the present exercise, we illustrate a case of joint space-conception, showing that design parts may be put together by establishing spatial relations among the parts. However, how to combine individual parts into integrated designs in a teamwork context is exploratory. There can be many other relations for the designers to discover. The spatial operations modelled in this chapter cannot be used to explain, for example, the design

integration occurred in the Seattle Fountain project shown in Chapter 2.

- OBJ3 was used as a simulation platform not a support platform. The impression that the two designers are modelling their design objects directly with the OBJ3 language should not be taken too seriously. As we know, in design, designers themselves employ language differently. There is no intention to say that OBJ3 is a ready-made language for collaborative design; rather the formal language was used to point to specifications of required properties of any conceivable support.

- The current scenario has not covered how what is shared by two designers might be extended so that a third designer, a late comer, perhaps, can participate as a member of a trio. We may think that through collaboration, A and B might be expected to extend their shared form of language expressions (adding spatial operations, objects etc.)—if C joins much later, how will C (even if C came to know the shared form at some earlier stage) get to know A and B's form, i.e., how does shared form extend?

Given all these 'unrealistic' impressions, does the current study imply anything useful at all? We believe that it does. It is not a simulation study of a real world but of a 'conceptual world'. The simulation is useful in two respects: first, it helps to define the underlying assumptions or boundaries of a formal approach to collaborative design more sharply; second, some pointers to system strategy can be drawn, if joint abstraction and use of shared integration schemas is to be developed into a realistic approach to collaborative design computing. The rest of this section is aimed to give more discussions of the aspects of a system strategy.

In general, we can think of a design environment developed to allow members of a design group to embark on two collaboration tasks: first, the construction of shared design constructs and operations with which domain design decisions can be integrated; second, the coordination of making local design changes. An overall view of the proposed components of a system strategy is provided in Figure 5.9

Joint abstraction of $\mathcal{SIS}$. To members of a design team, a shared integration schema ($\mathcal{SIS}$) is the common apparatus to project integrated design decisions as common images. It is important to recognise that a $\mathcal{SIS}$ has to be developed by group members on the basis of individual modelling methods. By jointly describing the (intended) relations among domain design elements, designers aim to acquire certain operations that can transform and interrelate parts of design objects with underlying

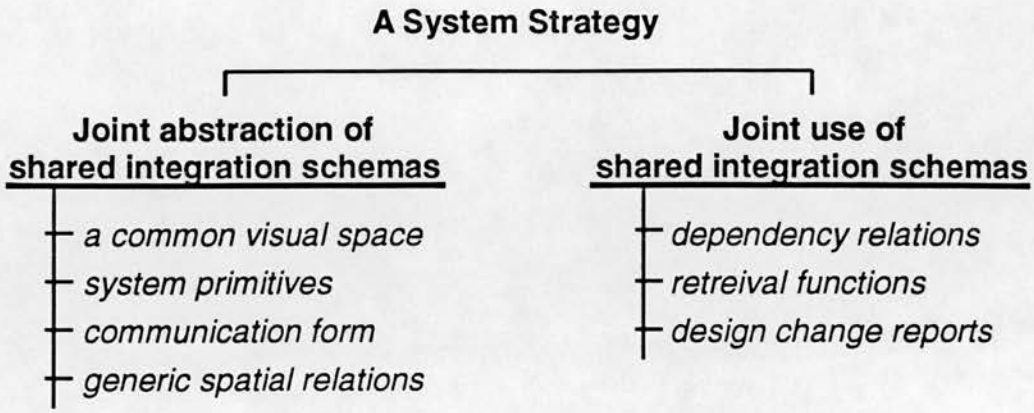


Figure 5.9. Components of a system strategy for supporting joint abstraction and use of shared integration schemas in collaborative design.

conceptual structures. The basic unit of a $\mathcal{SIS}$ is, therefore, a common (spatial) operation. Once established in a shared modelling space, a $\mathcal{SIS}$ can always be extended and repeatedly used by the team members. Since individual modelling methods are developed by designers at the time of designing, a system should not be expected to generate any integrative operations automatically. Instead, a system can be useful in facilitating the creative human teamwork in describing these operations. In this regard, our current study suggests the following system components:

(1) *A common visual space.* Participants' joint abstraction of shared design operations is motivated by their perceptions of potential interrelations among design instances. Therefore, prior to working out what operations are required, designers need to see each other's local design decisions in a shared workspace. This implies a shared visual space for displaying local design decisions in graphical form. Just for the purpose of displaying graphical stimuli, the common visual space is not required to accommodate any conceptual structures underlying the proposed design decisions.

(2) *System primitives.* In our OBJ3 simulation, a set of system primitives plays an essential role in the formulation of spatial operations and dependent relations of source designs. System primitives are useful when constructs defined in one domain are to be included in another existing or an emerging domain. Since individual modelling methods are built up by participants on the basis of system primitives, no extra cognitive load should be imposed on designers when using

these primitives in other occasions. However, the design of system primitives should not bear any references to specific objects in the real world if the users of the system work in multiple heterogeneous design worlds. Evidently, our current study cannot suggest what system primitives should be provided in a design environment. But given a limited number of system primitives, it should be possible that users can jointly set up new primitives as shared basic data types.

(3) *Communication form.* In the course of describing an integrative operation, designers need to collaborate on giving translations and inclusions of domain data types in a structured way. The structure can be derived from the descriptive scheme adopted by a system. There are other possible schemes for specifying spatial operations other than the algebraic one presented here. The idea is to provide the users with a 'communication form' on which each participant can fill in his or her bit of abstraction. Given a communication form filled in by one or more designers, a system is expected to generate the intended spatial operation.

(4) *Generic spatial relations.* A description of the conditions of applying a spatial operation is an integral part of describing the operation. As suggested by our simulation, a set of 'generic spatial relations' may be developed to ease condition description. The idea is that users can instantiate generic relations so that specific spatial conditions are specified (e.g., a generic spatial relation *x is located on y* is instantiated to *Opening is located on Edge*). Again, the question what generic spatial relations should be included in a system requires further research.

Joint use of $\mathcal{SIS}$. A shared integration schema is supposed to be installed and operated in a group modelling space. To use integrative operations for design integration, participants have to jointly provide combined source designs in a way 'recognisable' to the operations. The task of providing combined source designs legitimate to certain operations is a joint effort, since the sources must come from different design domains. Regarding the joint use of $\mathcal{SIS}$, we propose the following system components:

(5) *Dependency maintenance.* If dependency relations are declared among parts of source designs prior to integration, the relations need to be maintained so that changes can be affected by a system in one source design in response to the changes made in another. A set of spatial constraint satisfaction mechanisms can be useful in this aspect.

(6) *Retrieval functions.* What is required after integration is the articulation and distribution of the resultant components of an integrated design (e.g., the retrieval of

the enclosure or opening components from an integrated envelope). This implies that a system is able to perform certain retrieval functions when requested by the users. The processes of articulation and distribution can be done in a system-user interactive way, since any integrated design objects retain underlying formal structures recognisable both to the system and the users.

(7) *Design change reports.* Given the integration results articulated and distributed among individual modelling spaces, group members may carry out domain-oriented evaluations. In case some of the participants may find the consequences unsatisfactory, they set out to modify what is originally proposed in the source designs. Due to the system's dependency maintenance mechanisms, changes made in one source design may cause further changes to follow in other source designs. When a new state of combined source design is re-submitted to integration, undoubtedly, new states of resultant components will be generated. It would facilitate coordination and negotiation among group members, if a system can generate and deliver reports of design changes thus arising to the members involved in the same integration task.

Summary

Seen in the metaphorist pattern, common images are produced by members of a design group through their integration of local design decisions. To clarify the constraint of shared integration schemas classified previously, we set out a symbolic simulation of a simple shape construction exercise. The exercise starts with shape constructions in two separate modelling methods that are supposedly developed by two individuals; teamwork arises where *interpenetrations* between shapes of different conceptual domains are to be realised. Adopting an algebraic specification language, we aim to give an example of symbolic representation of a shared integration schema. Given the contents of the design exercise and the mathematics of the language, we extract the internal structures of the collaboration tasks simulated. In our current simulation, joint abstraction of shared integration schemas is modelled as cooperative specification of a set of spatial operations. Two other group tasks, provision of combined source designs and making changes in source designs, are analysed when we come to simulate the use of specified spatial operations. Drawing on the simulation results, we discuss the implications for a system strategy, if joint abstraction and use of shared integration schemas is to be developed as a form of collaborative design computing.

Chapter 6

The Substantiation of Common Generic Structures

This chapter presents an analysis of collaborative design activity, explaining how members of a design team collaborate to achieve integrated design on the basis of sharing and substantiating *common generic structures* with domain design developments. More examples of 'generic structures' taken from the history of architectural design are introduced, showing that teamwork can be based on participants' sharing some kind of generic objects. To define the issues to be better understood, a scenario of the *structuralist* approach to collaborative design is presented. The structuralist scenario is then given a situation-theoretical exposition. A set of constraints on collaboration are described, which serve to set down a structuralist logic of collaborative design: the necessity of maintaining a dual correspondence between the evolution of *common generic structures* and the development of *domain design solutions* distributed over several sites. Following the constraints identified, we discuss the basic requirements for computer-based tools to support the aspects of the structuralist approach.

6.1 The Sharing of Generic Structures

At the beginning of the thesis, we have indicated that our research aim is to develop a structural account of collaborative design in the field of architecture, which has the following features:

- *integration* — the teamwork has an ultimate goal of achieving a single integrated design that satisfies all participants' design requirements and judgements;

- *distribution* — the teamwork is undertaken by multiple individuals who are assembled as a working group because of their various design expertise;
- *evolution* — the teamwork itself is evolved in the interplay between *integration* and *distribution* (i.e., no fixed or pre-defined routes of integration or of distribution are given to designers in advance).

The 'structuralist' pattern characterised in Chapter 2 is just another approach to collaborative design that demonstrates the above features of teamwork. As shown in our case study of the Colonia Güell church project, the structuralist approach brings us to a crux of group design activity: some *common generic structures*, as we call it, are created collectively but viewed and substantiated differently by members of a design team. The term 'structuralist' is what we choose to name the observed pattern of teamwork in architectural design. But, as will be shown later, the special term is purely used to indicate the construction of 'structural objects' in group design modelling processes. It is not intended to relate to other notions or theories of 'structuralism', as developed by, e.g., Levi Strauss, Derrida and others.

Recalling the funicular model observed in the Colonia Güell church design, a model of that nature is an example of common generic structure. However, to show that the funicular structure is not the only instance ever invented in the history of architectural design, we now look at some other examples of common generic structures. In the following, we shall present a brief review of the evidence from the Greek *taxis* and its descendants.

The Greek taxis schema (or, the family of the taxis schemata) is basically a framework of 'subdivision' in an architectural composition. Tzonis and Lefaivre give the following characterisation of the taxis [TL87, p. 9]:

> "Taxis divides a building into parts and fits into the resulting partitions the architectural elements, producing a coherent work. In other words, taxis constrains the placing of the architectural elements that populate a building by establishing successions of logically organized divisions of space."

Figure 6.1 shows an example of the 'grid', or, more specifically, 'square' schema. The square framework is presented in an overall, general manner encompassing, initially, the whole area of the building (supposedly, a church). With its vertical and horizontal lines set up, it can be applied in a more specific way in controlling the position of the element of wall, defining the space for the nave and the aisle. An even more elaborate architectural plan can be produced on the basis of the grid pattern.

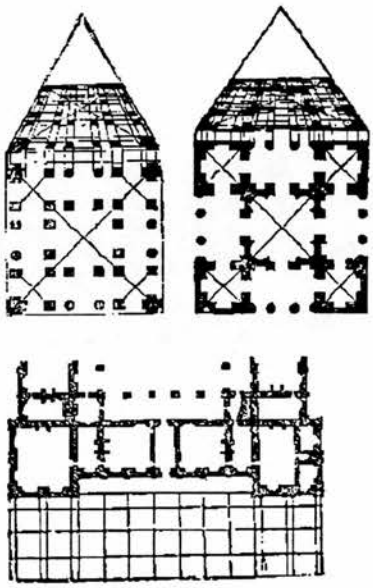


Figure 6.1. An example of square taxis for a church design, attributed to Serlio, 1691.

Another taxis pattern is the *polar* one. According to Tzonis *et. al.*, a polar taxis is a pattern where "One set of dividing lines forms concentric circles, while the other radiates from the common center of these circles." [TL87, p. 25]. Figure 6.2 shows examples of polar taxis the spatial subdivision is running in a circular manner.

The Greek taxis frameworks have been continuously used in the classical architectural composition, and its influence on later work can be seen at least as late as Durand and Guadet (see Figure 6.3 and Figure 6.4). As Tzonis and Lefaivre commented [TL87, p. 28]:

> "Toward the nineteenth century another way of specifying the divisions of a work became dominant: specification by an axis rather than by an outline. It is presupposed here that the architectural members of the section indicated by the axis are laid out—"balances"—around the axis according to bilateral symmetry."

What we see in the above examples of the taxis schemata is a family of frameworks or structural objects that are subject to various systems of *spatial* or *topological* logic. Clearly, this is a different category of *constraining forces* from that of the gravitational. It can be said that there are at least two categories of form-giving forces employed

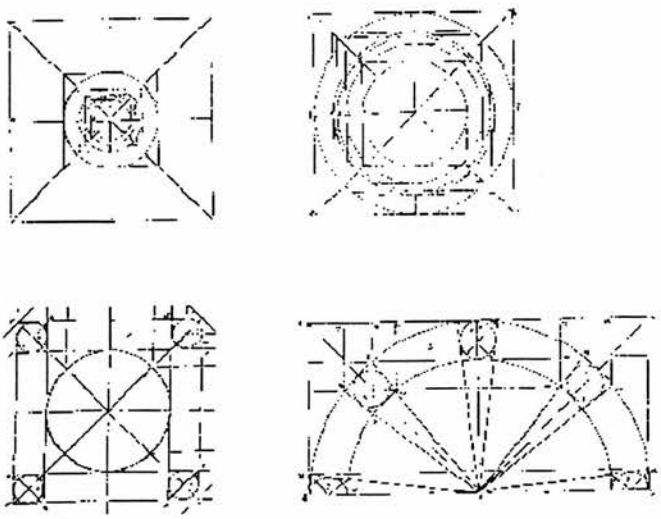


Figure 6.2. Examples of polar taxis schemata which partition building plans by means of contour and axis, attributed to Cousin, 1560.

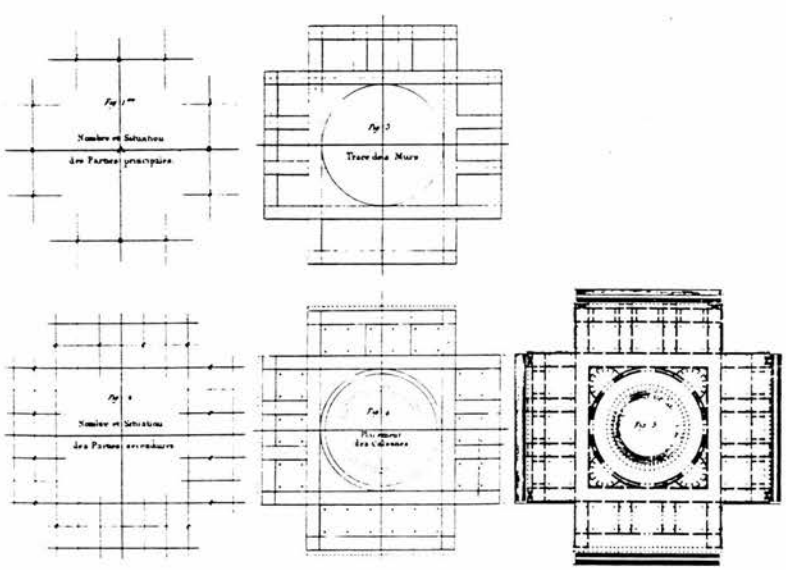


Figure 6.3. Examples of taxis schemata developed in the nineteenth century showing plans with subdivisions of plans embedded in them, from Durand's *Précis* 1802–1805.

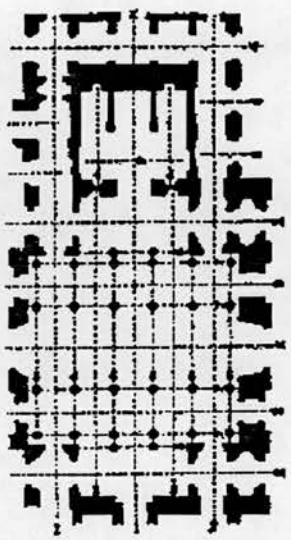


Figure 6.4. An grid pattern notation system showing the application of multiple taxis formulas on the same object by laying one over the other, from Guadet 1901–1904.

or employable in the field of building design: one is 'physical' (i.e., gravity, acoustics, light etc.), the other 'intentional' (i.e., some humanly devised spatial/shape grammars or schemata). As shown, both categories can act as systems of formal constraints in the construction of structural objects.

Why do we think that these structural objects have to do with an account of teamwork in design? First, these design objects are representations capable of demonstrating multiple design perspectives in their construction, manipulation, and substantiation. Second, these objects (2- or 3-dimensional) are themselves expressions of relations among constituting parts that may later contribute to form a larger whole. In contrast to the metaphorist approach, here we have 'emerging parts' instead of an emerging whole. Design of parts in the structuralist context is distributed among members of a group on the basis that an initial whole (a common generic structure, in our term) is presented.

When parts are developed locally to a certain extent, design judgements brought out by participating expertise will be given to these developments. These are actually interpretations of the *design consequences* of the existing generic framework. Since its being 'multi-viewpoint', a generic structure can be amended by whoever feels it

is inadequate or unsatisfactory. However, any changes made by one participant onto a generic framework will logically give rise to problems for other participants who share the framework. This is due to the action of a 'global' constraint system which will 'shape' the composition of a structure until a 'balanced' or 'equilibrium' state is reached. So, the structuralist proposition is that the sharing of a common generic structure gives rise to group interaction in collaborative design. If group members are to share a generic structure, they have to deal with any consequent problems, at least, inter-personally, if not cooperatively.

Having introduced the theme of sharing common generic structures in collaborative design, we may point out the following more specific questions to be answered:

(1) What are the representational elements or devices that enable a common generic structure to be constructed, manipulated, and changed from multiple perspectives brought by group members?

(2) What are the conditions that show whether a generic structure constructed in a shared workspace is 'shareable' or not among the contributors to its construction?

(3) How is the evolution of developing common generic structures participated by group members interrelated with the design developments pursued by individuals in separate domains?

(4) What do members of a design team have to know in order to initiate, or to respond to, group communications in the course of designing?

The remainder of this chapter is organised as follows. An overview of the structuralist approach is introduced in the next section. Given the structuralist scenario, the third section presents an analysis of the scenario based on our situation-theoretical framework. Within the framework, the constraints on collaboration are derived in the fourth section. Following the constraints identified, an informal specification of the requirements for computer support is presented in the fifth section. The chapter ends with a discussion of related work.

6.2 The Structuralist Scenario: An Abstract

Given the references from our observation of the sharing of generic structures as an approach to teamwork in architectural design and the historical case of funicular modelling in the Güell church project in particular, the 'structuralist' approach to collaborative design can be described concisely as follows:

At the inception of a design project, members of a design team work jointly in constructing a *group modelling space* for the modelling of a single structural or spatial framework as a *common generic structure*. There can be 'forces' introduced in the modelling space with which specific forms or shapes of the structure can be evolved. When a participant applies projective (derivative) devices onto a state of a common generic structure, *derivative structures* can be produced and imported to his or her *individual modelling space* that is set up and used by the individual for domain design purpose.

By taking derivative structures as *design referents*, group members carry out separate strands of *domain design developments*; they substantiate the imported derivative structures as parts of the generic structure into concrete and specific design expressions in their individual modelling spaces. In the course of elaborating domain design developments, some participants may be motivated, by seeing the design results in their working domains, to change parts of the derivative structures in use; to put the intended changes into effect, the individuals manipulate and modify the (corresponding) parts of the common generic structure. The changes thus proposed by one individual can subsequently cause further changes to be made in the derivative structures used by other participants.

6.3 An Exposition of the Structuralist Pattern

Employing the same methodological framework used in our analysis of the metaphorist pattern, this section presents a situation-theoretical exposition of the structuralist approach to teamwork in design. Likewise, our aim is to explore the constraints on the flow of information among the situation types classified in the structuralist scenario. After this, we will be in a position to describe systematically the requirements for computer support from what we know about the structuralist logic of collaborative design. To start with, a situation-theoretical analysis of the scenario, consisting of an action-space matrix, a classification of the situation types, and a scheme of the flow of information, is presented below.

6.3.1 Modelling spaces and modelling acts

Our classification of the modelling spaces for the structuralist pattern is the same as the metaphorist one. But in the structuralist context the general entities introduced

to modelling spaces are of a different nature:

GMS — The term 'group modelling space' (GMS) refers to a modelling space created by designers participating in a design project. One of the key functions of a GMS here is its use by all participants in modelling some kind of common generic structures (see more details below). In this regard, more specific qualifications of a GMS can be stated. A GMS may include the following basic types of components:

- *model constructs* — a collection of elementary objects that participants introduce to be used in the construction of a generic structure which is subject to the constraining forces deployed.
- *form-giving forces* or *constraints* — fields of physical or other kinds of forces that participants deploy to shape or deform parts of a generic structure. Given a constraining field in a GMS, all participants are concerned with a state of $\mathcal{CGS}$, as manifested in a configuration of instances of model constructs, to ensure it is in *equilibrium* or *satisfactory* to the forces or constraints applied.
- *manipulative operations* — operations that enable participants to displace, transpose, or aggregate etc. instances of various types of model constructs such that common structures can be created and evolved.
- *derivative operations* — operations that allow participants to perform certain spatial actions, such as sectioning, projecting, tracing, truncating, developing etc., so that 'secondary' structures can be derived.

IMSs — *Individual modelling spaces* (IMSs) are modelling spaces created and used by an individual for the development of his or her domain design solutions. In general, to carry out their individual design tasks, members of a design team may include the following components in their IMS set-ups:

- *individual object world* — a designer's design world, consisting of (domain-specific) notations and tools for coding, visualising, manipulating and evaluating design expressions.
- *derivative structures base* — an information space for storing, sorting, and displaying the secondary structures derived from an existing generic structure constructed in a GMS.

For the structuralist, we have the same entries of modelling acts, i.e., *abstraction, generation, interpretation,* and *modification.* Since no fundamental differences arise

Modelling Spaces \ Modelling Acts	*Abstraction*	*Generation*	*Interpretation*	*Modification*
GMS	*Shared Construction Set (SCS)*	*Common Generic Sturctures (CGS)*	*Derivative Structures (DS)*	*Changes in CGS or in SCS*
IMSs	*Individual Object Worlds (IOW)*	*Domain Design Expressions (DDE)*	*Local Design Judgements (LDJ)*	*Changes in DDEor in IOW*

Figure 6.5. An action-space matrix generating eight types of generic design states in the background of the structuralist pattern.

here, our general comments on these acts presented in Chapter 4 (see Section 4.4.2) are considered sufficient in the current context.

6.3.2 An action-space matrix

Given the modelling spaces and modelling acts specified above, an action-space matrix for the structuralist pattern is presented in Figure 6.5. As before, the matrix is constructed to reveal and classify the range of design states, which, in turn, gives rise to a list of the situation types in the structuralist scenario, inviting further explanations.

6.3.3 Situation types in the structuralist pattern

In consequence, we arrive at eight situation types which classify the structuralist approach. Guided by the format of an 'action-space-state' triple, more explanations are generated for each entry.

(1) [*abs*, GMS, $\mathcal{SCS}$] — participants of a design team perform *design abstractions* in a group modelling space resulting in a *shared construction set* ($\mathcal{SCS}$).

Recall the funicular modelling case. This is the situation where the basic model constructs (e.g., board, cord, weights, etc.) are introduced by group members. The constructs are the participants' abstractions of the design elements or factors that constitute the artefact to be built in the real world. We term the set of elements abstracted in a shared workspace for the modelling of generic structures (e.g., a funicular structure) the *shared construction set* ($\mathcal{SCS}$). Clearly, an

$\mathcal{SCS}$ is an information carrier at a low level, which can be shrunk or extended at some stage of group work.[1].

(2) [*abs*, IMSs, $\mathcal{IOW}$] — some individual performs design abstraction in his or her own modelling space resulting in an individual object world ($\mathcal{IOW}$).

This is the situation where elements of design representation are introduced by a designer working in a particular aspect of a design project. Similarly to the metaphorist equivalent, the term 'individual object world' is used to denote this type of design state. A designer's abstraction of an $\mathcal{IOW}$ can be seen as the point where an individual's design expertise enters, since an $\mathcal{IOW}$ is the working basis for its designer to contribute his or her part of substantiation of a common generic structure. Like the status of $\mathcal{SCS}$, an $\mathcal{IOW}$ is a low-level information carrier subject to changes whenever intended by its designer.

(3) [*gen*, GMS, $\mathcal{CGS}$] — one or more designers perform design generation in a group modelling space resulting in a state of common generic structure ($\mathcal{CGS}$).

Given a state of $\mathcal{SCS}$ is made available in a GMS, this is the situation where group members work (concurrently) in generating a configuration of generic structure. *Common generic structures* are 2-D or 3-D generic objects, representing, mainly, a kind of spatial framework or skeleton. As one of the essential features of the structuralist approach, the framework is constructed and can be subsequently used by all participants working in different domains of a design project. The following general properties of a generic structure can be observed:

- *Deformability.* A common generic structure is made of instances of model constructs that are connected in a field of physical forces or formal constraints. Being constructed and shared by all participants, a structure of this nature is meaningful and useful in revealing certain spatial forms or geometrical shapes. Changes in forces/constraints applied or in values of constructs may *deform* a structure into different states. The deformability entails that the construction of the generic objects is based on certain *physical* or *formal models* which behave in certain systems of constraint satisfaction or equilibrium of forces.

[1] We do not define the present situation type to include the realm of abstracting fields of constraining forces. In some cases, we believe, designers are not necessarily involved in representing a particular kind of form-giving force in a GMS. As in the funicular modelling case, designers simply know to use what is already available (i.e., the gravitational force).

- *Multiple-viewpoint.* Parts of a generic structure can be manipulated by participants from multiple points of view for different design reasons. The multiplicity is firstly achieved by participants' introducing types of model constructs that correspond to various 'perspectives' of design modelling (e.g. site, structure, enclosure, opening etc., in building design). Secondly, there are multiple ways allowed to assemble or detach model constructs while modifying parts of a structure. This multiplicity lies in a range of *connecting devices* that can be used to associate various types of model constructs introduced in the first place.
- *Derivability.* A state of generic structure can be applied with *derivative devices* as intended by any individual designers. Applied derivations can generate instances of derivative structures (see below) that can be further transported to individual workspaces for domain uses. The derivability allows participants to establish *referencing* relations between individual design developments and the evolution in their sharing a generic structure.

(4) [*gen*, IMSs, $\mathcal{DDE}$] — an individual performs design generation in his or her modelling space resulting in domain design expressions ($\mathcal{DDE}$).

Given an individual object world set up in an IMS, this is the situation where a designer generates design expressions specific to his or her domain design tasks at hand. The sharing of a common generic structure is only half of the structuralist story. As a project developed, final design products often go well beyond the design of generic skeletons. An equally important part is concerned with how the generic structures can be substantiated with more specific substances or properties. In this respect, domain design expressions are the outcomes from the specialisation processes carried out by individuals with various design expertise.

Being different from the metaphorist, the generation of domain design solutions here is closely associated with another type of design state—derivative structures (see below). The difference lies in that, in the structuralist approach, all local design expressions are made on the basis of the structures or frameworks derived from a $\mathcal{CGS}$. That is, designers generate ($\mathcal{DDE}$) by taking derived structures as underlying design referents. For this connection, we introduce a different type of design state to denote the difference.

(5) [*int*, GMS, $\mathcal{DS}$] — a designer's performing design interpretation (of a state of common generic structure) in a group modelling space resulting in *derivative structures* ($\mathcal{DS}$).

Given the set of derivative devices made available in a GMS, this is the situation where a designer applies the devices provided onto parts of a $\mathcal{CGS}$ and produces $\mathcal{DS}$ as information of secondary order (e.g., the inverted photographs taken from the exterior or interior of the funicular model). As observed, *derivative structures* are basically 2-D or 3-D pictorial objects representing derivative spatial frames or skeletons. Images of $\mathcal{DS}$, once imported into individual workspaces, can serve the individuals as referents in generating domain design expressions. We attribute this activity as an interpretation because how and where to apply derivative devices involves a designer's knowledge about domain design tasks. In a sense, this is similar to saying that to designers with different design interests, a common generic structure might have different meanings as to how it might be used.

To look at the status of $\mathcal{DS}$ in a further depth, in the course of developing domain designs, a designer may feel the need to manipulate parts of $\mathcal{DS}$ acting as the underlying design referents. However, the manipulation has to be largely indirect. Since instances of $\mathcal{DS}$ are 'frozen' images in its nature, they cannot be manipulated in parts but only as a whole. To effect changes in the referents of this nature, corresponding changes have to be affected in the $\mathcal{CGS}$, so that a new state of $\mathcal{DS}$ containing the intended changes can be re-derived.

(6) [*int*, IMSs, $\mathcal{LDJ}$] — a designer's performing design interpretation in his or her own modelling space resulting in local design judgements ($\mathcal{LDJ}$).

When a domain design expression is made in an IMS, this is the situation where the designer gives interpretation of the expression arrived at. Since each $\mathcal{DDE}$ is generated in relation to what $\mathcal{DS}$ is underlaid, and any instance of $\mathcal{DS}$ is a derivative of $\mathcal{CGS}$, the resultant $\mathcal{DDE}$, as viewed and judged by its creator from a particular design perspective, is a *design consequence* of $\mathcal{CGS}$. A participant may be motivated by his or her state of $\mathcal{LDJ}$ to carry out design modification both in IMS and in GMS.

(7) [*mod*, GMS, $\Delta\mathcal{CGS}$ and/or $\Delta\mathcal{SCS}$] — one or more designers' performing design modification in a group modelling space resulting in changes in common generic structures and/or in the shared construction set.

When a state of $\mathcal{LDJ}$ arises in the head of a participant, he or she will get access to GMS to change the attributes of instances of model constructs (e.g., moving a cord from one place to another, or distributing weights in a different manner), so that undesirable design consequences (as manifested in $\mathcal{DDE}$) can be

removed. Or, more fundamentally, he or she may introduce new model constructs to the existing $\mathcal{SCS}$, so that a new aspect of modelling $\mathcal{CGS}$ can be opened up. In both cases, this type of situation will give rise to communication and coordination among group members (see more discussion in the constraint analysis below).

(8) [*mod*, IMSs, $\Delta\mathcal{DDE}$ and/or $\Delta\mathcal{IOW}$] — a designer performing design modification in his or her own modelling space results in changes in domain design expressions and/or in individual object worlds.

When a new state of $\mathcal{DS}$ is imported in an IMS, this is the situation where an individual executes intended changes in an existing $\mathcal{DDE}$. As new design expressions are unravelled with reference to the underlying $\mathcal{DS}$, the designer may have further interpretations regarding the modified state of $\mathcal{CGS}$, and forms his or her latest $\mathcal{LDJ}$. So the structuralist story goes on until, perhaps, no participants in a design group express the need to change any bits of $\mathcal{CGS}$ and $\mathcal{DDE}$s.

6.3.4 Information flow in the structuralist pattern

Putting the above situation-theoretical constructs into an order that follows the procedural features described in the scenario, we may construct a chart of information flow in the structuralist pattern. To better illustrate the flow, a diagram showing a scheme of the information flow in the structuralist pattern is provided in Figure 6.6. The chart starts with the formation of a shared construction set upon which the generation of common generic structures is based. Derivative structures are derived from the generics and imported to a number of distributed individual modelling spaces. With reference to the underlying derivative structures imported, domain design expressions are made in individual object worlds. Domain-oriented interpretations of the expressions result in local design judgements which, in turn, motivate changes to be made in the underlying derivative structures. The design changes thus initiated in each domain may result in further changes to be affected or propagated in parts of a common generic structure.

6.4 Constraints on Structuralist Collaboration

In searching for the constraints on the flow of information, we have located two places where constraints on the flow of design information give rise to group interactions for accomplishing joint design modelling tasks (see where the digits '1', '2', are located in Figure 6.6):

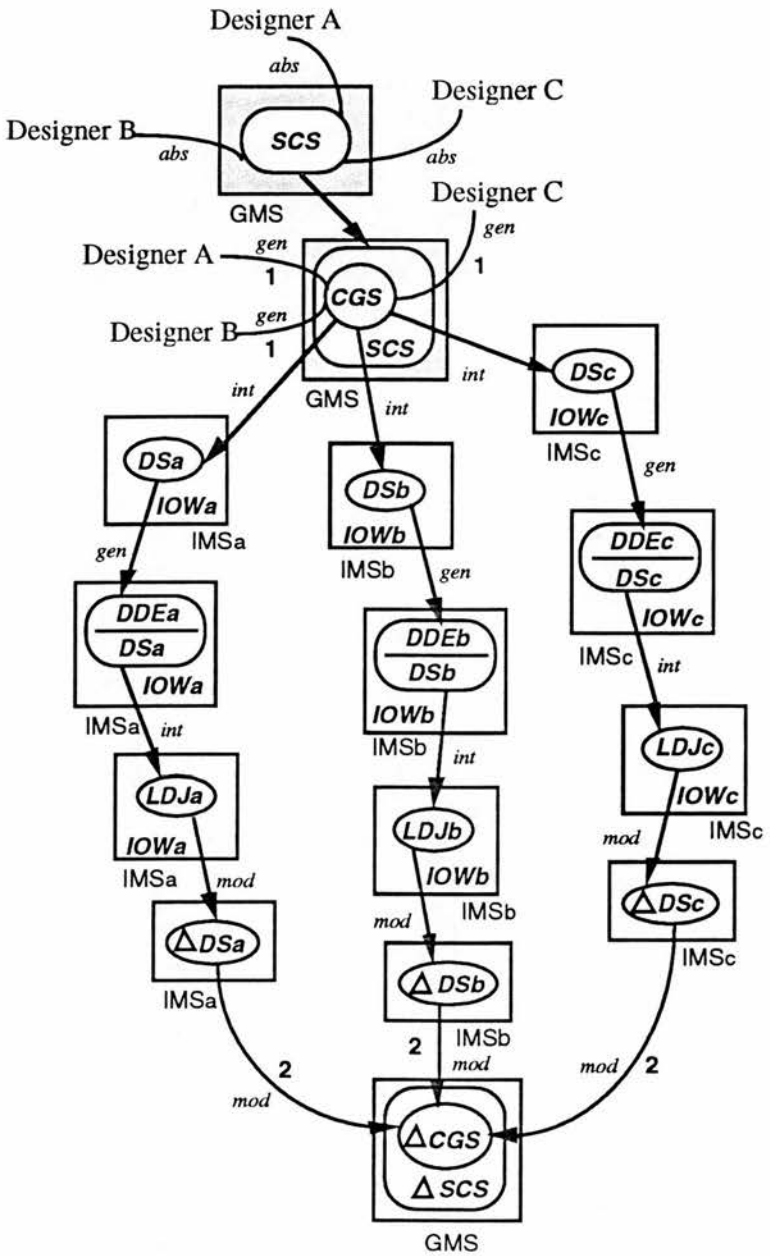


Figure 6.6. A scheme of the flow of information among the situation types classified in the structuralist approach. (Note that the number of designers shown on this scheme is only an assumption; in theory, the number can be scaled up to any group size.)

(1) from $\mathcal{SCS}$ to $\mathcal{CGS}$ — i.e., from a situation (type) where a shared construction set is put forward into a group modelling space to a situation where a common generic structure is generated in a group modelling space.

Question: Given a shared construction set introduced in a GMS, how do instances of the constructs come together in forming a generic structure that is *shareable* to (all) members of a design team?

In the structuralist approach, an important issue is how to achieve and evolve a common design framework that can serve the participants to develop domain-specific substantiated designs. Since the framework has to be shared by all group members through a design project, the property of being 'shareable' of a common generic structure is a critical indicator of the continuing of teamwork. The shareability of $\mathcal{CGS}$ reflects the status of common understanding and judgement achieved and maintained by team members. In certain circumstances, a $\mathcal{CGS}$ may become not shareable, hence teamwork cannot continue, due to the following conditions:

- *deformability* — a $\mathcal{CGS}$ may not be deformable in a GMS because an *equilibrium* state of the structure under construction cannot be reached, i.e., the generic structure collapses under the constraining forces applied;
- *multiple-viewpoint* — a $\mathcal{CGS}$ may not be accessible to some participants in the course of modelling because of the lack of certain types of model constructs or connectors;
- *genericity* — a $\mathcal{CGS}$ may not be usable to some members because its derivative structures are not generic on an appropriate level to serve the purposes of domain-oriented design substantiation.

Among the above three factors, the availability of *model connectors*, we would argue, is the most crucial one. Types of model constructs may be brought into play by participants in an arbitrary way in the first instance. Clearly, if two types of model constructs are introduced by two different designers, to make use of instances of the constructs in building a structure, a way of making the connections must be found, and this can be a 'design experiment' jointly carried out by the two designers. We shall characterise this group activity as 'making and issuing model connectors'. A collection of model connectors is a kind of connective device that can be used to join instances of different types of model constructs in relation to an acting field of form-giving forces. Here we are thinking

of the 'hooks', 'clippers', 'jointers' etc., shown in the funicular modelling case (see Section 2.2.3). Once a set of connectives is made available in a GMS, the deformability, multi-viewpoint, and genericity of a $\mathcal{CGS}$ can then be built and tested.

Given the role of connectives as this, we propose the availability of model connectives as the first structuralist constraint[2] on collaboration[3]:

$$\textit{Constraint } 1: \ (\mathcal{SCS} \rightsquigarrow \mathcal{CGS}) \Longrightarrow \Sigma \ (\textit{Connectives})$$

(2) from *design changes in derivative structures* ($\Delta\Sigma\ \mathcal{DS}_i$) to *design changes in a common generic structure* ($\Delta\ \mathcal{CGS}$) — i.e., from a situation where one or more participants bring up proposals for design changes to be made in the derivative structures in their uses to a situation where design changes are reflected in parts of a generic structure.

Question: What is involved for the design changes in derivative structures as intended by one or more participants to be finally realised?

To better understand the above question, we need a more thorough discussion of the 'consistency' relations among $\mathcal{CGS}$, $\mathcal{DS}$, and $\mathcal{DDE}$.

1. $(\mathcal{CGS})R_d(\mathcal{DS})$ — Derivative structures are *derived from* a state of a common generic structure. Therefore, $\mathcal{CGS}$ always stands in a relation, denoted as R_d, to $\mathcal{DS}$. In theory, the type of relation R_d can be characterised in terms of, for instance, the derivative devices (methods) used and the spatio-temporal locations (relative to the $\mathcal{CGS}$ in a GMS) of applying the devices.
2. $(\mathcal{DS})R_f(\mathcal{DDE})$ — Domain design expressions are constructed by participants *with reference to* underlying derivative structures. Therefore, $\mathcal{DS}$ always stand in a relation, denoted as R_f, to $\mathcal{DDE}$. Examples of the type of relation R_f can be, for instance, geometrical or spatial referencing, or material substantiating.

[2]Note that the term 'constraints' in the context of "constraints on information flow" here has a different domain of discourse from that of 'constraints' in the context of "form-giving forces as design constraints" as described in Section 6.3.1. Constraints on information flow are the systematic relations we use to characterise group interaction in collaborative design; while constraints in the realm of design modelling are the systematic relations employed by the designers to shape design objects. Clearly, the former is more general than the latter.

[3]A general reading of the terms and notations used here in describing the structuralist constraints on collaborative design is provided in the Glossary—see Appendix A.2, page 208, and Appendix A.3, page 209.

3. $(\mathcal{CGS})R_d(\mathcal{DS})R_f(\mathcal{DDE})$ — Given the above two relations, a more complex relation among $\mathcal{CGS}$, $\mathcal{DS}$, and $\mathcal{DDE}$ can be formed among the three design states. In two aspects, this complex relation describes the problem of maintaining the consistency of information flow:

$$\mathcal{CGS} \rightsquigarrow \mathcal{CGS}' \ (1) \ ; \quad \text{(some participant's act causing state changing in } \mathcal{CGS})$$
$$\mathcal{DS} \rightsquigarrow \mathcal{DS}' \ (2) \ ; \quad \text{(because a } R_d \text{ is in operation)}$$
$$\mathcal{DDE} \rightsquigarrow \mathcal{DDE}' (3) \ ; \quad \text{(because a } R_f \text{ is in operation)}$$

Or the information flow can be the other way around

$$\mathcal{DDE} \rightsquigarrow \mathcal{DDE}' \ (1) \ ; \quad \text{(some participant's act causing state changing in } \mathcal{DDE})$$
$$\mathcal{DS} \rightsquigarrow \mathcal{DS}' \ (2) \ ; \quad \text{(because a } R_f \text{ is in operation)}$$
$$\mathcal{CGS} \rightsquigarrow \mathcal{CGS}' \ (3) \ ; \quad \text{(because a } R_d \text{ is in operation)}$$

Regarding the issue of consistency among different design states, we may propose two other structuralist constraints on collaboration in the following.

Suppose, at some design stage, a participant (say, Designer A) decides to make some changes in $\mathcal{DS}s_A$ (i.e. the set of derivative structures used by Designer A) to maintain (or validate) an intended domain design solution. Consequently, A's changing $\mathcal{DS}_A$ leads to a changing state of the $\mathcal{CGS}$ which is shared by other participants (say, Designers B and C). Owing to the deformability of the $\mathcal{CGS}$, the derivative structures, $\mathcal{DS}_B$ and $\mathcal{DS}_C$, used by B and C respectively may get changed in order to maintain the derivative relations introduced previously. This gives rise to at least two circumstances calling for communication among A, B and C:

$$\textit{Constraint } 2: \ \Sigma\Delta\ \mathcal{DS}_i \rightsquigarrow \Delta\ \mathcal{CGS} \Longrightarrow \textit{Coordination}$$

Coordination is involved if A's changing $\mathcal{DS}_A$ is seen in $\mathcal{CGS}$ and judged desirable by B and C, as they inspect the consequent states of their own derivative structures. Under this circumstance, B and C need to coordinate A's proposals by making further developments in their domain design solutions in respect of the changed $\mathcal{DS}_B$ and $\mathcal{DS}_C$.

$$\textit{Constraint } 3: \ \Sigma\Delta\ \mathcal{DS}_i \rightsquigarrow \Delta\ \mathcal{CGS} \Longrightarrow \textit{Negotiation}$$

Negotiation is involved if A's changing $\mathcal{DS}_A$ is judged undesirable by B and/or C, as they inspect the differences occurred in the changing states of $\mathcal{DS}_B$

and/or $\mathcal{DS}_C$. Under this circumstance, A needs to negotiate with B and/or C by either dropping completely the intended changes in $\mathcal{DS}_A$, which requires A to develop his or her domain design in a different direction, or requesting B's and/or C's suggestions of the extent to which the changes in $\mathcal{DS}_A$ are acceptable.[4]

To summarise, we may spell out the structuralist logic of collaborative design from the constraints identified above: participants in a design project collaborate with one another by building a generic structure in a shared work space, which provides a common framework capable of linking domain design developments undertaken by participants in a distributed manner. A deformable and generic structure can be built and evolved from multiple viewpoints, if and only if a set of *model connectives* is made available. *Coordination* is involved if one designer's intention to change parts of a generic structure is acceptable to other team members; otherwise *negotiation* among participants for making design changes is necessarily involved until the *shareability* of the generic structure is regained.

6.5 Issues in Supporting Structuralist Collaboration

Given the structuralist constraints on collaboration arrived at the above, we now move onto a discussion of the representation and communication requirements for tools to support collaborative design. Like our treatment of the metaphorist requirements, a formal and complete requirement specification for the structuralist approach will not be presented. Instead, we aim to identify areas of prospective computer support which have not been fully addressed by current research in collaborative drawing support tools, and to suggest the connections with related research done in other areas.

(Issue 1) Support for the Construction of a GMS

As clearly revealed in the structuralist scenario, collaborative design begins with the construction of a common modelling space. Given the initial demand, a computer-based design environment may have to provide, in the first instance, representation support in the users' construction of a GMS.

[1.1] *Representation of multiple viewpoints.* — More specifically, this requirement can be subdivided into the representation of two kinds of model objects:

[4]Note that A's receiving the suggestions made by B and/or C is same as how B and/or C may recognise A's intention expressed in changing $\mathcal{DS}_A$; the cognitive basis for A to do so is, again, the deformability of $\mathcal{CGS}$ and the relations between $\mathcal{CGS}$ and $\mathcal{DS}$.

[1.1.a] *Types of model constructs* — Participants working on various aspects of a project need to introduce types of model constructs that they consider pertinent representations of design elements. Different viewpoints in a GMS may be better represented by various types of *model constructs*. Instances of model constructs can interact with form-giving forces or constraints applied in a GMS and exhibit certain behaviours of deformation.

[1.1.b] *Types of model connectors* — When types of model constructs are introduced by participants, model connectors, the devices to connect or disconnect instances of the constructs, are essential. Types of connectors are used by participants to define and effect ways of manipulating parts of the common structures for various reasons. Note that model connectors are *neutral* objects in a sense that they do not represent any specific design elements in the real world.

[1.2] *Representation of constraint system for shaping $\mathcal{CGS}$.* — Design participants are not expected to build up, computationally, a common constraint system by themselves for modelling $\mathcal{CGS}$, since this demands highly technical knowledge. It would be a task for system engineers to develop computational models that can interact with instances of model constructs and connectors introduced by participants. The design of constraint systems of a GMS can be of the following nature:

[1.2.a] *General constraint systems* supporting physical (or, more broadly, environmental) laws such as gravity, thermal energy, or acoustics etc.

[1.2.b] *Specific constraint systems* supporting intentional laws such as particular systems of spatial or shape grammars.

[1.3] *$\mathcal{CGS}$ is pictorial and generic.* — Representation of $\mathcal{CGS}$ requires to be graphical and, at the same time, generic for the following two reasons:

[1.3.a] In serving *all* members of a design team as a common (global) representational medium, $\mathcal{CGS}$ is essentially *pictorial*, or, at least, *diagrammatic*. This implies that the construction of $\mathcal{CGS}$ has to be based on graphical objects so that all participants of different backgrounds can feel relatively easy to be familiar with.

[1.3.b] $\mathcal{CGS}$ is essentially generic in order to be enriched or refined to different levels of specificity. Therefore, its representation requires, perhaps, a higher order of genericity to support the following flow of information:

$$\mathcal{CGS} \overset{instantiation-of}{\longrightarrow} \mathcal{DS} \overset{substantiation-with}{\longrightarrow} \mathcal{DDE}$$

(Issue 2) Support for the Construction of IMSs

In teamwork, a participant's development of domain design solutions is not less important than that of common structures. To carry out more technical modelling tasks, participants need to work with *personal* workspaces which are not necessarily known and accessible to others. The problem is how to have a system capable of interacting with a user and generating an IMS which he or she thinks pertinent to the design tasks at hand. This requirement gives rise to the following sub-issues:

[2.1] *Representation of individual object worlds* — This includes, firstly, a set of personal design constructs for generating and manipulating domain design expressions, secondly, domain- oriented constraint systems employable in shaping domain design developments.

[2.2] *Support for the construction of $\mathcal{DDE}$ with reference to $\mathcal{DS}$* — The spaces for constructing $\mathcal{DDE}$ is required to be *overlapped* or *juxtaposed* with the spaces for holding $\mathcal{DS}$ as design referents.

[2.3] *Support for the construction of $\mathcal{DDE}$ by substantiating $\mathcal{DS}$ with domain design elements (substances)* — This is a user's need for direct use of $\mathcal{DS}$ imported from a GMS. The type of construction process involves enriching or refining $\mathcal{DS}$ into $\mathcal{DDE}$ filled with more domain design details.

(Issue 3) Support for Coordination and Negotiation

The representations in a GMS and multiple IMSs discussed above can be termed the 'infrastructural' supports for the users' setting up group as well individual workspaces. In addition to infrastructures outlined, there are system requirements of more dynamic features to be considered. The third issue is concerned with system ability to support the constraints of *coordination* and *negotiation*:

[3.1] *Detection of state change in $\mathcal{CGS}$* — It is clear to us that $\mathcal{CGS}$ is a dynamic object subject to participants' manipulations from different viewpoints. It is the evolving of a common structure that gives rise to the

dynamism of teamwork. For a design environment to fit into the dynamic situation, it has to be concerned with the facts about state change in $\mathcal{CGS}$. But how do we define such a state change?

[3.1.a] A state of $\mathcal{CGS}$ is defined by a two or three dimensional deployment of instances of model constructs and connectors under the influence of a global constraint system activated in a GMS.

[3.1.b] A state change in $\mathcal{CGS}$ can therefore be defined as a change in (parts of) an existing deployment (or, a better word, configuration) resulting from a net effect of some participant's or participants' modelling actions together with the constraint influence.

A system's ability to keep track of state changes in $\mathcal{CGS}$ lies in whether the system can generate information about the configuration differences between two $\mathcal{CGS}$ states given at a time. This bit of information is essential for the system to trigger further communicational mechanisms, such as the maintenance of R_d and messages delivering for users' maintaining R_f (see below). Seen in this requirement, a detection mechanism, so to speak, is needed.

[3.2] *Maintaining the relation R_d in $(\mathcal{CGS})R_d(\mathcal{DS})$* — In developing domain design solutions, participants need to extract derivative structures from a state of $\mathcal{CGS}$ as design resources or references. Since the state of $\mathcal{CGS}$ may keep changing, it is a useful support for participants if a system can inform the users in a timely way of the changing states of $\mathcal{DS}$ in use, arising from state change in $\mathcal{CGS}$. This requires a system to keep a record of the relation between $\mathcal{DS}$ and $\mathcal{CGS}$ and compute updated states of $\mathcal{DS}$ whenever $\mathcal{CGS}$ gets changed. Apart from the state of $\mathcal{CGS}$, two representations are necessarily involved in a system's maintaining the relation R_d:

[3.2.a] *Representation of derivative operations* — To derive a $\mathcal{DS}$, users require to perform certain *spatial operations*, such as *projecting, subdividing, or slicing* etc., upon $\mathcal{CGS}$. Taken as a bit of information, a *derivative action* thus consists of the performer and the spatial operation performed.

[3.2.b] *Representation of location of deriving* — The information about the time and position (relative to $\mathcal{CGS}$ modelled in a GMS) in which a derivative action takes place is also relevant in keeping a R_d.

[3.3] *Message delivery for maintaining the relation R_f in $(\mathcal{DS})R_f(\mathcal{DDE})$* — Standing from a domain design perspective, a participant shall perceive his or her development of domain design solutions as design consequences in relation to a state of $\mathcal{CGS}$. By judging the resulting development, any participant may well be motivated to make changes in $\mathcal{DDE}$. This kind of design change activity gives rise to a second dynamism in the course of teamwork. As explained before, there exists the systematic relation, R_f, between $\mathcal{DS}$ and $\mathcal{DDE}$. Given a change in $\mathcal{DDE}$ desired by some individual, a R_f will not be sustainable if state changes in $\mathcal{DS}$, and hence in $\mathcal{CGS}$, are not reflected correspondingly.

A usable collaborative modelling environment should, therefore, not only allow for participants to freely make changes in $\mathcal{DDE}$ in their IMSs, but also assist the individuals in dealing with the problem of maintaining R_f. To support this communication need, two functionalities are considered necessary:

[3.3.a] *Detection of state change in $\mathcal{DS}$* — A detection mechanism similar to that of detecting $\mathcal{CGS}$ state changes is needed. But the detection functions need to be installed locally as IMSs may be distributed over a number of separate working sites.

[3.3.b] *Sending the change message to GMS* — When a state change $(\mathcal{DS} \rightsquigarrow \mathcal{DS}')$ is computed, a message is sent to GMS for activating corresponding state change in $\mathcal{CGS}$.

When GMS receives and processes the message sent from IMSs, a change in $\mathcal{CGS}$ will be implemented by the system, resulting in $\mathcal{CGS}'$. Owing to the mechanism of maintaining R_d described earlier, further messages (containing the information about $\mathcal{DS} \rightsquigarrow \mathcal{DS}'$) being sent from GMS to IMSs shall naturally follow so that other participants involved shall be informed. The detection and message delivery mechanisms described here seems to suggest a *local management agent* be set up in an IMS which is the sole information space for the agent to serve.

[3.4] *Communication channels for resolving conflicts manifested in $\mathcal{CGS} \rightsquigarrow \mathcal{CGS}'$* — If the coordination constraint, as described previously on page 149, cannot be satisfied, negotiation among the individuals involved in the disagreement is needed to resolve the conflict. Since the situation is a highly non- deterministic one, a system is not expected to automatically detect the arising of a conflict and resolve it. In principle, this should be left to the

participants to decide if coordinating or negotiating is needed. In coordination, there is no need for participants to express individual judgements of the state of $\mathcal{CGS}$, and corresponding changes in $\mathcal{DDE}$ shall be carried out in IMSs separately.

More problematically, in negotiation, participants need to express disagreement to one another[5]. This demands a system to provide users with communication channels with which they can discuss, directly or indirectly, and resolve the differences in recognising the state of $\mathcal{CGS}$ until the sharability is re-established among members of a design team.

6.6 Related Research

To investigate the possibility of computer-supported collaborative design, we have started from a study of the structuralist approach to teamwork in architectural modelling. By carrying out an informal analysis of the structuralist scenario, a classification scheme that explains the constitution of collaborative design activity is presented. Guided by an examination of the properties of the types of representation and the systematic relations among them, a logic of collaboration in teamwork is found. The constraints spell out what is involved when members of a design team co-work on the substantiation of a common generic structure with heterogeneous design developments in a distributed manner. Following the constraint presentation, we then give a discussion of the basic requirements for prospective computer support.

For the purpose of drawing up a promising strategy for a further exploration, we have some readings from other researchers. In relation to our current enquiry, the following collection of research references are of particular interest:

1. In their search for what makes research on Computer Supported Cooperative Work (CSCW) a unique field, Schmidt and Bannon propose a general conceptual framework for CSCW (see [BS91, SB92]). In particular, they argue that the priority of computer support for group working should be aiming at two aspects: support for the *articulation work*, and support for the *construction of a common information space*. Our current study of the structuralist (and the metaphorist) approach seems to be positive about the two supporting aspects as pointed out by the Schmidt-Bannon framework. But in the light of the structuralist and the metaphorist experiences, we would argue that computer support

[5] Again, to use the negotiation situation described on page 149, this is to say that B and/or C must find a way to let A know that A's intention in making the change in $\mathcal{CGS}$ is not acceptable.

for the *interaction* between articulation and construction should be added as the third dimension.

2. Based on analyses of organisational problem solving in scientific communities, Leigh Star derives the concept of *boundary objects* and suggests the concept would be an appropriate data structure for Distributed Artificial Intelligence (DAI) [Sta89]. Star identifies four types of boundary object which are considered as a major method of solving heterogeneous problems. Notably, the properties of boundary objects bear a close relation to those of our common generic structures [Sta89, p. 46]:

   > "Boundary objects are objects that are both plastic enough to adapt to local needs and constraints of several parties employing them, yet robust enough to maintain a common identity across sites. They are weakly structured in common use, and become strongly structured in individual-site use."

   We suggest that the $\mathcal{CGS}$ in our case can be another candidate for a type of boundary object to be used in collaborative design but with a generic-specific structural adaptation instead of a weak-strong one. Though the general properties of boundary objects are researched, no computational representations of the objects have been proposed.

3. Research on computer graphics models, which can respond in a natural way to applied forces or constraints, has shown us the technical possibilities of representing $\mathcal{CGS}$ graphically in computers. In particular, three research results worth noting: the theory of elasticity was employed by Terzopoulos *et al.* to construct elastically deformable models [TPBF87]; Witkin and others explored the representation of (geometrical) constraints as energy functions that behave like forces pulling and deforming parts of the model into place [WFB87]; three force-based constraint methods were explored by Platt and others to add several desirable properties into flexible models [PB88]. It certainly remains to be seen how the kinds of graphics model achieved above can serve in a collaborative design context, satisfying the demands for being generic and manipulable for multiple design purposes. As for representing design constraints on a smaller scale, Gross and others developed constraint-based design environments as separate specialised design 'labs' (see [GEAF88] for detail). We see this work as a precedent experiment to the setting of private constraint systems in distributed IMSs.

4. System research and developments on computer-supported human communication in cooperative work have presented two distinct approaches: one is in favour of supporting informal interaction among users through the design of *shared virtual workspaces*; the other focuses on supporting structured group interaction mediated by *communication protocols*. In the realm of synchronous group drawing, as shown in our system survey in Chapter 3, research prototypes of shared drawing systems have demonstrated a range of technological possibilities to re-create face-to-face communication where users are actually separated geographically.

   Along with the second line, several computer-based coordinating protocols have been implemented. The building of these mechanisms is mainly based on representation, in a formal language or system, of either particular work procedures or a body of (often domain-specific) knowledge involved in the design tasks. Recalling the *NETWORK* design environment reviewed earlier, the idea is to implement such a system to test how useful the *structured* domain design knowledge is going to be in the hands of computer network designers [FGL+92, RS92]. Not appearing in our survey, Bond and others develope a set of *rules of interaction*, arising from "an organisationally agreed sequence of commitment steps", to model the collaboration among specialists in aircraft design [Bon89, BR92].

   It remains questionable, however, whether the knowledge-based approach to the design of computer-based coordinating mechanisms can satisfy the needs of less stabilised group practice in design. The encapsulation of specific knowledge about artefacts or procedures can be problematic to collaborative design that demands unique solutions to every single project. Obviously, it is not very sensible to design a collaborative design environment centred on the funicular structure shown in the Colonia Güell church project; as we know, there are always innovative building structures being developed.

Summary

Following the structuralist pattern outlined in Chapter 2, this chapter develops an account of collaborative design activity which is featured by group members' joint substantiation of common generic structures. To show that the funicular structure seen earlier is not the only case, we present further evidence of generic frameworks from a collection of taxis schemata. Structural objects of this nature play an interesting role in collaborative design. They can be created, manipulated, and evolved by all members of a design team from different design viewpoints. Once constructed in a

shared workspace, common generic structures can function as communication frameworks upon which participants interact with each other in the course of developing domain-specific designs.

To unravel what conditions the structuralist approach, we carry out a situation-theoretical analysis of an abstract scenario. A scheme of information flow among the situation types classified indicates two transitions where group interaction might have to occur. One is the generation of a generic structure from a shared construction set, where participants need to jointly experiment with *model connectives* so that instances of various model constructs can be linked in a way responsive to the action of form-giving constraints. Much more dynamic group interaction can occur when any of the group members attempts to make design changes. Motivated by domain design judgements, one or more participants may go on changing parts of a structure which may cause further changes to follow in other members' work. The continuation of teamwork is therefore constrained by *coordination* or *negotiation.*

We derive three issues to be addressed when computer-based group design environments are to be developed. In the representation aspect, general schemes can be developed to support the construction of the various components of individual as well as group modelling spaces. In the communication aspect, mechanisms such as *state change detection* and *relation-maintaining message delivery* can be looked at to facilitate design coordination and negotiation. A potential technological basis for developing collaborative design computing of this nature can be found by interrelating some of the related researches done in computer graphics, distributed artificial intelligence, and collaborative drawing support tools.

Chapter 7

Collaborative Substantiation of Common Generic Structures: A Simulation

In this chapter a computational simulation of the structuralist approach to collaborative design is presented, focusing on the sharing of common generic structures ($\mathcal{CGS}$). The sharing of $\mathcal{CGS}$ among members of a design team is understood as if a state of $\mathcal{CGS}$ can be substantiated with more specific design expressions given by the team members working in various domains. Two design examples are described; the first, as a general case, illustrates design as a process of substantiating generic outlines with detailed designs, the second example, by extending the first case into a collaborative one, describes a case of joint design substantiation. The two examples are intended as (group) design tasks for a symbolic simulation. Again, the algebraic specification language OBJ3 is chosen as a simulation apparatus, for which we give a brief introduction to its formalism that supports parameterisation and instantiation. Given the formal mechanisms introduced, we present and explain the results of a simulation of the two design examples. Seen from our current simulation, the implications for developing joint substantiation of common generic structures as a strategy of collaborative design computing are discussed in the conclusion.

7.1 More on the Sharing of $\mathcal{CGS}$

As a general proposition of collaborative design, it can be said that designers develop certain kinds of design objects that are shareable among group members over the life

time of a design project. By classifying the design objects of this nature and analysing the relations among them, we have tried to describe the representation and communication requirements with an eye on potential areas of computer support. Our requirement study presented in the previous chapter indicates that if computer-based tools are to be developed to support collaborative design with the structuralist features, the enabling of a group of users to create and share their common generic structures seems to be a legitimate system goal. Certainly, the goal of supporting distributed heterogeneous substantiation of common generic structures presents considerable challenges to system development.

The structuralist pattern explained in the previous chapter suggests that designers can communicate and coordinate with each other on the basis of some sort of 'structural objects'. Because of the sharing of these structural objects among designers with different expertise, the term 'common generic structures' ($\mathcal{CGS}$) was introduced to denote this teamwork feature. Several properties of the structural objects are observed to be relevant to the facilitation of teamwork in design; these include genericity, deformability, and multiple-viewpoint. However, the group dynamics of the structuralist approach to collaborative design cannot be explained solely by $\mathcal{CGS}$ itself. As analysed, a $\mathcal{CGS}$ is always situated in a bigger context consisting of other categories of representation objects. Just to recap what we have come across previously:

- *Common generic structures* made of model constructs and connectives representing a common ground at its lowest level;
- *Field of form-giving forces* introduced to represent or simulate certain systems of design constraints;
- *Derivative structures* ($\mathcal{DS}$) acquired and used by individuals as underlying design outlines or references in domain-oriented design developments;
- *Domain design expressions* ($\mathcal{DDE}$s) resulted from participants' distributed design substantiation showing the consequences of an existing common generic structure.

In addition to the above range of representation objects, we also have the following types of design operations classified, which, when applied, can cause design information to flow among the objects:

- Construction and manipulation of a $\mathcal{CGS}$;
- Derivation or projection of $\mathcal{DS}$s from a $\mathcal{CGS}$;
- Domain substantiation of $\mathcal{DS}$s into $\mathcal{DDE}$s.

Given these key elements of the structuralist requirements, how can we demonstrate these elements in a computing platform, so that a direction for the development of collaborative design computing can be clarified further?

Like our treatment of the metaphorist requirements, a more narrowing-down analysis of the sharing of $\mathcal{CGS}$ at a lower level is presented in this chapter. The present analysis is based on an OBJ3 simulation of simplified design examples[1]. The purpose of the simulation is twofold: first, to give an example of symbolic representation of design as giving substances to underlying generic structures or outlines; second, to see if any internal structures of collaboration tasks of joint substantiation of $\mathcal{CGS}$ can be proposed. To our purpose, the design examples introduced in this chapter are therefore of two kinds. We first look at design as substantiating generic structures or outlines in general (i.e., considering the design exercise undertaken by an individual in the first instance). The general example is then extended into a teamwork context.

The remainder of the chapter is organised as follows. In the next section, two examples of design as substantiating generic structures in an individual and a group context are described, delineating what is to be simulated. To better present and explain the simulation results, a short introduction to the concepts and methods of parameterisation and instantiation in OBJ3 is provided in the third section. The fourth section presents our current simulations of the design examples. Finally, pointers to a system strategy for developing joint substantiating of generic structures as a form of collaborative design computing are discussed.

Like our previous simulation of a simplified version of the metaphorist approach to collaborative design, OBJ3 code is frequently introduced to present our simulation of the structuralist approach. However, a reading of the code may be skipped if the technical details embedded in the simulation appear overly design- or otherwise, system-oriented. Explanations of the quoted code are given, which help to illustrate the concepts of design modelling and of the structuralist approach to collaborative design simulated. An overview of what has been described in the OBJ3 simulation is provided in the concluding discussions.

[1] A simulation of, for instance, the funicular modelling case, or, of the Greek taxis schemas would certainly appear much more realistic; but the computational complexity involved might appear not so relevant to the current scope and aim of this chapter.

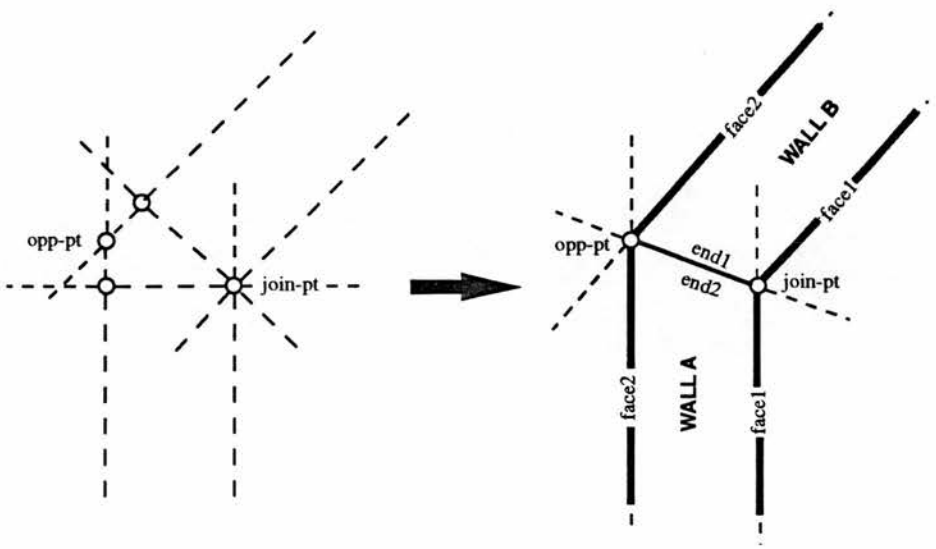


Figure 7.1. An example of design as substantiation of a generic outline. Walls (in thick solid lines) can be seen as being 'built upon' the underlying construction lines (in dotted lines and small circles). (After Figure 7.9 of *Designing in Words and Pictures*, Bijl 1989)

7.2 Some Design Examples

We shall look at design examples of two kinds. The first is a case of a design exercise supposedly undertaken by a single individual. The purpose of introducing this example is to illustrate more precisely the main points of design as substantiating generic structures. The individual case is then extended into a teamwork context, where two (imagined) designers participate in the sharing and substantiating of a generic design outline.

7.2.1 A general case of design substantiation

In illustrating how the design philosophy of the MOLE system is related to design practice in architecture, Bijl gives a working example of "joining walls" [Bij89, pp. 197–205]. Figure 7.1 shows a graphic depiction of the design problem and task.

To better see the relation of the end-on wall junction design to our purpose of giving an example of design substantiation, Bijl's description of the MOLE working example is quoted below [Bij89, p. 200]:

> "... drawings of walls start off as variants of rectangles, with long segments

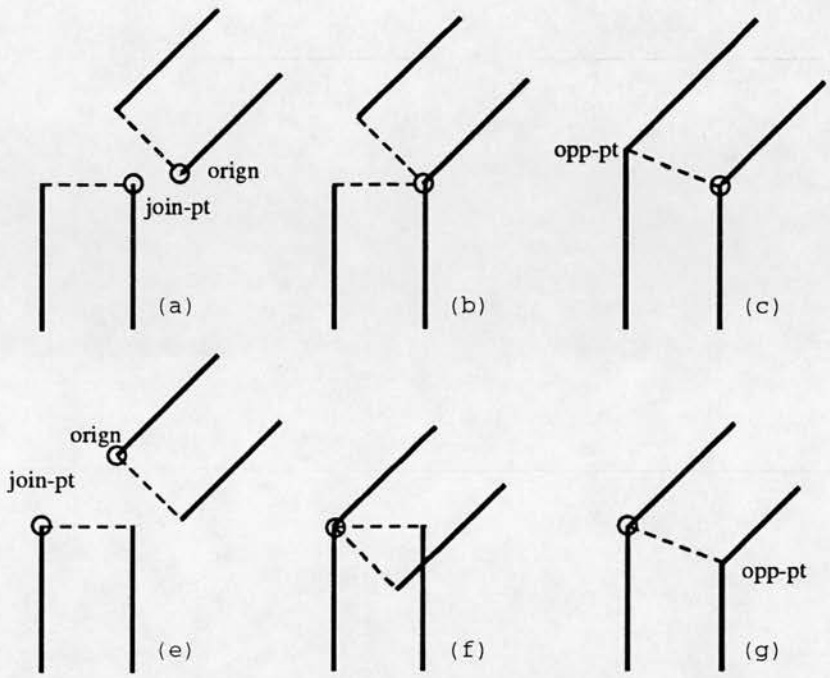


Figure 7.2. The 'stretching' and 'contracting' operations for dealing with the problem of wall junctions. (After Figure 7.8 of *Designing in Words and Pictures*, Bijl 1989)

> that are stretchable and short segments of fixed length. The long segments are referred to by named parts of walls, 'face1' and 'face2', and the short segments are named 'end1' and 'end2'. The system knows that walls have faces and ends and it knows of wall types (external, internal etc.) which govern the lengths of ends, and wall materials (brick, block, etc.) which can be associated with the drawings of walls. When any two walls are to be joined, the drawing depicting one wall is moved to the drawing depicting another wall, the latter already being fixed within some arrangement of wall."

The diagrams given by Bijl in Figure 7.2 show the number of operational steps needed to form junctions when combining two joining walls. The steps (a) to (c) can be seen as the 'stretching' operation, which produces an 'opposite point' `opp-pt` on the convex faces of the joined walls; the steps (d) to (f) the 'contracting' operation, producing an `opp-pt` on the concave faces of the joined walls.

Intuitively, the correspondence between the wall joining example and our thinking of design as substantiation of generic structures can be drawn in terms of the following:

1. *Outline constructions.* This example reflects the 'top-down' approach to drawing that can be seen in many design fields. Design is started by first setting up its "controlling construction lines" and then developed gradually, by filling in details. The construction of outlines that describe a general configuration or framework, therefore, corresponds to the notion of constructing generic structures.

2. *Transformation rules.* The stretching and contracting operations of the example are in fact transformation rules. These rules say that if an initial configuration meets certain conditions then it will be transformed into a new configuration by executing certain 're-drawing' procedures. In our view, these rules correspond to the notion of a field of form-giving forces.

3. *Detailed designs.* In the example, more specific design information (e.g., choices of different types of walls) is added onto or associated with the outline design constructed earlier. This can be seen as if a generic outline is substantiated with domain- specific design substances represented by further graphic and linguistic notations.

7.2.2 An example of joint design substantiation

We now give the second design example by extending the above wall junction design example into a teamwork context. That is, we consider a case of collaborative design where the design of a wall junction is now undertaken jointly by two designers (Designer A and Designer B, say). Figure 7.3 illustrates how we think of a simple example of joint design substantiation.

It should be admitted that this example in its current form may appear unrealistic and superficial. However, by pointing out the following correspondence between the bits of the design example and the parts of the structuralist framework, we consider this design example is indeed a case for simulation:

- *Joint construction of a $\mathcal{CGS}$ in a GMS.* We assume that in a modelling space shared by Designer A and Designer B, there is a construction set (construction line and point) with which the designers can construct any arbitrary configurations. There are also the 'stretching' and 'contracting' operations applicable to initial configurations. The wall junction design starts with Designer A's and/or Designer B's putting up instances of construction lines (shown in [a]), onto which an operation is applied, aiming at a resultant generic outline (in [b]);

- *Deriving $\mathcal{DS}$s from a $\mathcal{CGS}$.* Two derivative structures, $\mathcal{DS}_1$ (in c]) and $\mathcal{DS}_2$

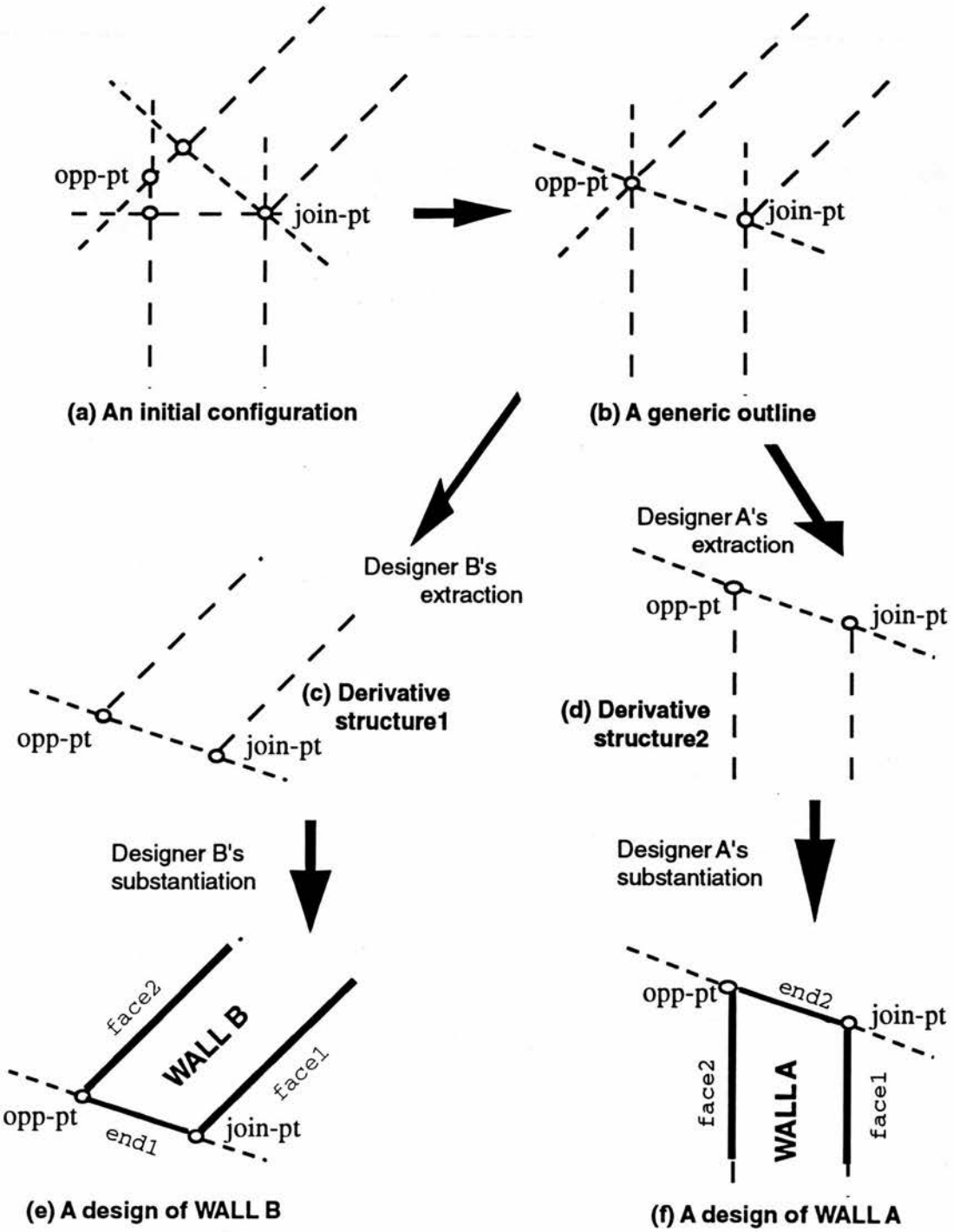


Figure 7.3. An example of joint design substantiation of a common generic outline shared by two designers.

(in [d]), are extracted from the resultant generic outline by Designer A and B respectively;

- *A local design substantiation.* $\mathcal{DS}_1$ is substantiated into *Wall-B* (in [e]) by Designer B in B's individual modelling space;
- *Another local design substantiation.* $\mathcal{DS}_2$ is substantiated into *Wall-A* (in [f]) by Designer A in A's individual modelling space;
- *Judgements on $\mathcal{DDE}$s giving rise to making changes in $\mathcal{CGS}$.* On seeing the design consequence of the join-end of *Wall-A* and *Wall-B*, Designer A decides to increase the thickness of *Wall-A* by moving *face1* (in $\mathcal{DS}_2$) to its right and *face2* (in $\mathcal{DS}_2$) to its left.
- *The shareability of a state of a $\mathcal{CGS}$ is in question.* When the design changes intended by Designer A are reflected in the initial configuration, a new state of a resultant generic outline is yielded. Regarding this, Designer B may or may not agree with the new state of $\mathcal{DS}_1$, which suggests some *follow-up* changes to be made in the domain of *Wall-B*.

7.2.3 What is to be simulated?

For the two design examples above, we intend to give the following interconnected parts of design substantiation both in the general and in the collaborative cases a symbolic simulation, using the language OBJ3:

1. a modelling space which provides some elementary constructs for describing initial design configurations, and the 'stretching' and 'contracting' operations applicable to any given configurations;
2. the production of design outlines as generic structures from transforming initial configurations according to whether the stretching or the contracting operation is applied;
3. the substantiation of an underlying design outline with some domain-specific substances that are described in an individual modelling space;
4. the extraction of derivative structures from an outline design that is assumed to be constructed and shared by the two designers;
5. the joint substantiation of a shared design outline (a junction) with domain-specific substances specified in two individual modelling space;

6. the situation where changes in one derivative structure give rise to changes in a common design outline, which, in turn, suggest the changes to follow in another derivative structure.

Having explained what is to be simulated, a short introduction to the simulation apparatus we are going to use in the modelling of the design examples is provided in the next section.

7.3 Parameterisation and Instantiation in OBJ3

As a continuation of the introduction to the language OBJ3 presented in Chapter 5, we now introduce briefly the concepts of *parameterisation* and *instantiation* of the language. Since these programming concepts are constantly applied and referred to in our simulation of the design examples, a familiarity with these formal concepts is considered necessary prior to a presentation of the simulation itself. The formal approach of *parameterised programming* was first developed by computer scientists for the purpose of achieving maximal and successful *reuse* in software engineering. As Goguen and Winkler put it [GW88, p. 21]:

> "The basic idea of parameterised programming is a strong form of *abstraction*: to break code into highly parameterised mind-sized pieces. Then one can construct new programs from old modules by instantiating parameters and transforming modules."

The formal mechanisms provided by OBJ3 to implement parameterised programming include the following:

1. *requirement theories* — In OBJ3, a *theory* defines the interface of a parameterised module, that is, the 'structure' and 'properties' required of an actual parameter (see later) for meaningful instantiation. A theory can be parameterised by containing one or more interfaces declared in other theories.

2. *parameterised modules* — A parameterised module is an abstract data type containing one or more requirement theories as its interfaces. In general, a parameterised module is an abstract function which can be applied to specific data types when it is instantiated with actual parameters.

3. *actual parameters* — Actual parameters in OBJ3 are executable modules in which given expressions can be evaluated. Modules like `POINT` and `LINE` shown in Chapter 5 are examples of actual parameters seen from the parameterised programming point of view.

4. *views* — It is possible for a theory to be satisfied with a module in more than one way. A view is a piece of code expressing that a certain actual parameter satisfies a certain theory in a certain way. Basically, a view describes a *binding* of the signature of an actual parameter to the signature of a requirement theory.

5. *instantiation* — An instantiation of a parameterised module is the construction of a new executable module by giving the OBJ3 system the theories, views, and actual parameters associated with the parameterised module. In Goguen and Winkler's words [GW88, p. 21], "instantiation of a parameterised module with an actual parameter, using a particular view, yields a new module."

To illustrate the above basic concepts of parameterised programming in OBJ3, we now give an example made of several modules respectively: a requirement theory, a parameterised module, an actual parameter, a view and an instantiation of a new module.

An example of parameterised programming in OBJ3 Consider a simple design task of moving an object along with a reference framework, a (base) line in a two-dimensional space, say, in a way such that a reference (central) line across the object is aligned with a given base line in relation to a particular 'anchoring' point on the base line (see Figure 7.4). The object may be a square, rectangle, or a circle. In the following, we present a theory of 'alignment' and a parameterised 'anchoring' function. An actual parameter of 'rectangle' is chosen for the time being; a view from the alignment theory to the object type rectangle is provided. A new function of 'anchoring rectangle' is then instantiated, which can deal specifically with transposing a rectangle to a specified location on a base line.

A theory of alignment The key elements of our current alignment theory are: we have an object made of four components to be moved around; any of the four components can be specified by two points in a two-dimensional space. There is also a reference line associated with the object. Bearing these general elements in mind, the signature body of the theory can be specified in terms of the following *constructors, predicates*, and *selectors*[2]:

[2]In talking about specification of abstract data types, or data abstraction, Harrison classifies three categories of functions—constructors, predicates, and selectors [Har89, pp. 14–15]; rather recently, Willis and others suggest another categorisation of constructors, modifiers, and observers [WP92, p. 31].

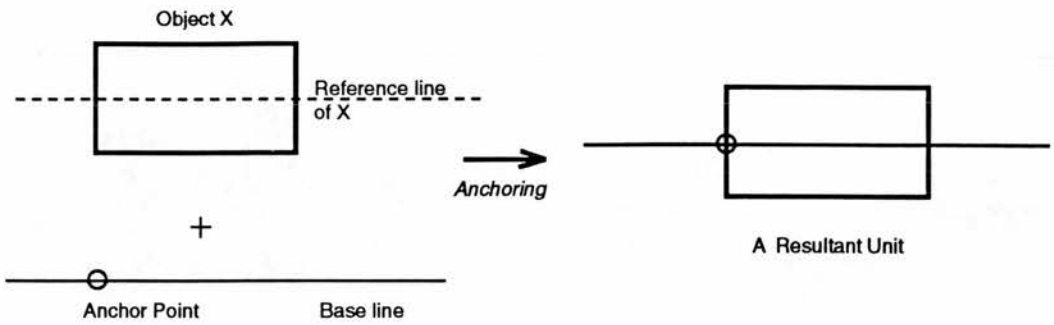


Figure 7.4. Anchoring an object on a base line with reference to an anchor point if the object is a rectangle.

```
th ALIGN is protecting LINE .
  sort Point .   *** the element
  sorts Comp1 Comp2 Comp3 Comp4 . *** four components
  sort Object .  *** an object of 4-component
  sort RefLine .   *** reference line
  subsort RefLine < Line .
  op comp1 : Point Point -> Comp1 .
  op comp2 : Point Point -> Comp2 .
  op comp3 : Point Point -> Comp3 .
  op comp4 : Point Point -> Comp4 .
  op select-P1-Comp1 : Comp1 -> Point .
  op select-P2-Comp1 : Comp1 -> Point .
  op select-P1-Comp2 : Comp2 -> Point .
  op select-P2-Comp2 : Comp2 -> Point .
  op select-P1-Comp3 : Comp3 -> Point .
  op select-P2-Comp3 : Comp3 -> Point .
  op select-P1-Comp4 : Comp4 -> Point .
  op select-P2-Comp4 : Comp4 -> Point .
  op object : Comp1 Comp2 Comp3 Comp4 -> Object .
  op find-refline : Object -> RefLine [memo] .
  op is-refline : Point Point -> RefLine .
  op select-Comp1 : Object -> Comp1 .
  op select-Comp2 : Object -> Comp2 .
  op select-Comp3 : Object -> Comp3 .
  op select-Comp4 : Object -> Comp4 .
  op select-P1-RefLine : RefLine -> Point .
  op select-P2-RefLine : RefLine -> Point .
```

The operational semantics of the above operations are specified in a number of

equations which can be found in Appendix D.1.1. In short, the theory specifies nothing in particular, but three requirements of the abstract object `Object`: (1) "`Point` is the element", (2) "The object is an object of four components", and (3) "For each object, there is a reference line can be found".

Parameterised anchoring To specify the generic function of anchoring an object on a base line, we start with the inputs of the function: an object, an anchoring point, and a base line. The intended output would be the translated object with reference to the base line and the anchor point. Most importantly, the generic function takes the alignment theory as an interface which has already given a basic definition of what an object is. The purpose of the parameterised module is to give the data abstraction of a 'Unit' which is the end product from the operations of translation and anchoring.

```
obj ANCHOR[X :: ALIGN] is
  sort Unit .
  sorts Anchor-pt BaseLine .
  subsorts BaseLine < Line .
  op anchor-pt : Float Float -> Anchor-pt .
  op baseline : Point Point -> BaseLine .
  op anchor : Object Anchor-pt BaseLine -> Unit .
  op isUnit : Object BaseLine Anchor-pt -> Unit .
  op is-anchored-on : BaseLine -> BaseLine .
  op at-anchor-pt : Anchor-pt -> Anchor-pt .
  op translation : Object Anchor-pt -> Object .
  op x : Anchor-pt -> Float . *** selector of anchor point
  op y : Anchor-pt -> Float . *** selector of snchor point
```

A full specification of `ANCHOR[X]`[3] is provided in Appendix D.1.2.

"Rectangle" as an actual parameter Now an actual object "rectangle" is considered. A rectangle is said to have two short sides and two long sides. Two reference lines are defined in terms of the central lines running through the short and long sides respectively. Apart from the selectors, important constructors are the operations to find the two reference lines of any given rectangle.

[3]In OBJ3, the symbol "`::`" in `ANCHOR[X :: ALIGN]` is a system-defined character, which is used to declare the formal parameters of a parameterised module. A parameterised module can have more than one parameters.

```
obj RECTANGLE is protecting SEGMENT .
  sort Rectangle .
  sorts S-side1 S-side2 L-side1 L-side2 .
  sort SC-Line .  *** Central Line on short side
  sort LC-Line .  *** Central Line on long side
  sort Area .   *** Area of a rectangle
  subsorts S-side1 S-side2 L-side1 L-side2 < Segment .
  subsort SC-Line LC-Line < Line .
  subsort Area < Float .
  op rec : S-side1 S-side2 L-side1 L-side2 -> Rectangle .
  op sc-line : Rectangle -> SC-Line .
  op lc-line : Rectangle -> LC-Line .
  op area : Rectangle -> Area .  *** compute the area of a rec
```

For a full specification of RECTANGLE, see Appendix D.1.3.

A view from Align to Rectangle To instantiate a parameterised module meaningfully, a view is needed. A view is constructed by mapping (all) the sorts and operators specified in the requirement theory to those (not necessarily all) in the actual parameter. In our current example, this is a mapping between the signature of the theory ALIGN and (parts) of the signature of RECTANGLE. The following is one of several other possible mappings:

```
view ALIGN-REC from ALIGN to RECTANGLE is
  sort Comp1 to S-side1 .    sort Comp2 to S-side2 .
  sort Comp3 to L-side1 .    sort Comp4 to L-side2 .
  sort Object to Rectangle .  sort RefLine to SC-Line .
  op comp1 to s-side1 .       op comp2 to s-side2 .
  op comp3 to l-side1 .        op comp4 to l-side2 .
  op object to rec .
  op select-Comp1 to select-S-side1 .
  op select-Comp2 to select-S-side2 .
  op select-Comp3 to select-L-side1 .
  op select-Comp4 to select-L-side2 .
  op select-P1-Comp1 to s-P1-S-side1 .
  op select-P2-Comp1 to s-P2-S-side1 .
  op select-P1-Comp2 to s-P1-S-side2 .
  op select-P2-Comp2 to s-P2-S-side2 .
  op select-P1-Comp3 to s-P1-L-side1 .
  op select-P2-Comp3 to s-P2-L-side1 .
  op select-P1-Comp4 to s-P1-L-side2 .
  op select-P2-Comp4 to s-P2-L-side2 .
```

```
  op find-refline to sc-line .
  op is-refline to is-SC-Line .
  op select-P1-RefLine to select-P1-SC-Line .
  op select-P2-RefLine to select-P2-SC-Line .
endv
```

Clearly, another legitimate mapping is `sort RefLine to LC-Line`, `op find-refline to lc-line`, and `op is-refline to is-SC-Line`, since there are two ways of constructing a reference line for a rectangle. Different views will give rise to different instantiations, resulting in different new modules.

Making a function for anchoring rectangle We now have the theory `ALIGN`, the parameterised module `ANCHOR[X :: ALIGN]`, the actual parameter `RECTANGLE`, and the view `ALIGN-REC`. When all these modules are compiled in the OBJ3 database, we are in a position to instantiate a new module capable of modelling the transposition of a rectangle onto a base line. This can be done by giving OBJ3 the following module expression: `make ANCHOR-REC is ANCHOR[ALIGN-REC] endm`

The `make` expression above is to instruct the OBJ3 system to produce a new module by instantiating `ANCHOR[X]` with the view `ALIGN-REC` filled in. As a result, a new module, named `ANCHOR-REC`, is generated in the OBJ3 database. Within this module, we can construct instances of rectangles, base lines, and anchor points. By applying the anchoring operation onto an assembled source instance, we shall get the result whereby a given rectangle is translated and anchored on a base line.

Reduction in `ALIGN-REC` First, a straightforward construcion of a rectangle.

```
red rec(s-side1(point(4.6,7.5),point(4.6,4.2)),
        s-side2(point(9.8,7.5),point(9.8,4.2)),
        l-side1(point(4.6,7.5),point(9.8,7.5)),
        l-side2(point(4.6,4.2),point(9.8,4.2))) .
rewrites: 0
result Rectangle: rec(s-side1(point(4.6,7.5),point(4.6,4.2)),
    s-side2(point(9.80,7.5),point(9.80,4.2)),l-side1(
    point(4.6,7.5),point(9.80,7.5)),l-side2(point(4.6,
    4.2),point(9.80,4.2)))
```

Then, an application of the function `anchor` onto the above rectangle together with an anchor point and a base line.

```
red anchor(rec(s-side1(point(4.6,7.5),point(4.6,4.2)),
               s-side2(point(9.8,7.5),point(9.8,4.2)),
               l-side1(point(4.6,7.5),point(9.8,7.5)),
               l-side2(point(4.6,4.2),point(9.8,4.2))),
           anchor-pt(5.7,-3.2),
           baseline(point(4.6,-4.2),point(9.8,-1.60))) .
rewrites: 658
result Unit: isUnit(rec(s-side1(point(5.7,-1.55),point(
    0.50,-4.85)),s-side2(point(10.9,-1.55),
    point(5.7,-4.85)),l-side1(point(5.7,-1.55),
    point(5.7,-1.55)),l-side2(point(5.7,-1.55),
    point(0.50,-4.85))),is-anchored-on(baseline(
    point(4.6,-4.2),point(9.80,-1.60))),at-anchor-pt(anchor-pt(
    5.7,-3.2)))
```

Using the above example, we hope to have introduced the basic scheme of parameterised programming in OBJ3. Due to the lack of space, an exploration of its full potential is not allowed here. The 'scaling-up' of our anchoring example can be said to be twofold: (1) given a theory and a parameterised module, a range of new modules can be instantiated by providing either different views to the same actual parameter (as mentioned earlier) or to different actual parameters (for instance, a view from **ALIGN** to **SQUARE**, or, to **RHOMBUS**, if a model of square or rhombus is provided); (2) a parameterised module can have more than one requirement theory as its interface (e.g., putting another theory of 'rotation' into the parameterised anchoring function).

The design of OBJ3 has delivered some powerful mechanisms for building up larger systems or models from existing smaller ones in a stepwise manner. Other programming languages, for instance, Ada [oD80] and Modula-2 [WP92], also provide similar but weaker formal support for parameterised programming. With the building of requirement theories, a designer can not only specify the syntax but also the semantics of the interfaces of a parameterised module. However, this is not a place to discuss further details of the formal techniques. Our purpose is to use the OBJ3 platform for a simulation of the design examples given in Section 7.2.

7.4 From Generic Outlines to Wall Junctions

In the design examples described earlier, we consider that design in general is a process of gradually giving specific substance to some generic structures. This can happen in individual as well as in group design practice. Our presentation of OBJ3 simulations of the two examples in this section is to demonstrate how the conceptual framework

of (joint) design substantiation can take a computational form. Through these simulations, we hope to draw some implications for a system strategy of collaborative design computing. The simulation is presented in two parts: first a general individual case, then an extended joint one.

7.4.1 A simulation of design substantiation

Our first design example on page 162 shows a description of the task of making a wall junction. In our view, the task can be carried out in two different design worlds: one is the 'platonic' world of construction lines and points, the other, a more materialistic world of walls. With parameterised programming in OBJ3, we first model the different design worlds in two separate modules. We then show how to make connections between the two worlds by constructing a theory, a parameterised function, and a view, so that a demonstration of design as substantiation of generic outlines can be made as clear as possible.

The entire simulation of the example is carried out in seven steps. First, a basic *construction space* is simulated by providing a set of constructs for making initial configurations subject to operations defined in the space. The outputs from the operations are *generic outlines* in the sense that they bear nothing in particular other than the platonic elements specified in the construction space. A module providing a set of constructs and operations for describing things in a world of *walls* is given, which will act as an actual parameter later on. We then present a *theory of substantiation* which specifies what elements in a generic outline are to be given specific contents. A parameterised function, which takes our theory of substantiation as the interface, delivers an abstract regime in which a generic outline is put together with design substances into substantiated design. To launch an instantiation, a view from the requirement theory to the actual world of walls is prepared. Given the theory, the parameterised module, and the view, an instantiation is performed, resulting in a new construction space where design substantiation can be supported. Finally, some results of running the programmes are shown.

Generic configurations and outline Following what was said in the wall junction design example, the basic construction space centres on the descriptions of two types of constructs: joining and fixed rectangles[4]. A meeting of these objects gives rise to a certain type of *junction*, depending on what operation (i.e., stretching or contracting)

[4]For simplicity, we consider only three-sided joining and fixed rectangles in the present simulation.

is applied. The initial rectangles and the resultant junction are complex objects, each of which is made of simpler constructs. To be able to cover the full range of making junctions from rectangles, we end up with various types of construction points (origin, joint-point, opposition-point by stretching, and opposition-point by contracting), and several types of segments that constitute the rectangles and targeted junctions. The following just shows some of the main ingredients of the signature of the module `JOINING`:

```
obj JOINING is protecting SEGMENT .
   sorts Join-rec Fixed-rec Translated-rec .
   sorts Origin Join-pt S-opp-pt C-opp-pt .
   sorts Join-seg1 Join-seg2 Fixed-seg1 Fixed-seg2 .
   sorts S-j-seg S-f-seg F-seg J-seg C-j-seg C-f-seg .
   sorts Join-side Fixed-side .
   sorts M-j-seg M-f-seg .
   sorts S-Junc C-Junc . *** Junction by stretching, contracting
   subsorts Origin Join-pt S-opp-pt C-opp-pt < Point .
   subsorts Join-seg1 Join-side Join-seg2 < Segment .
   subsorts Fixed-seg1 Fixed-side Fixed-seg2 < Segment .
   subsort M-j-seg M-f-seg < Segment .
   op origin : Float Float -> Origin .
   op join-pt : Float Float -> Join-pt .
   op join-seg1 : Origin Point -> Join-seg1 .
   op join-seg2 : Point Point -> Join-seg2 .
   op join-side : Origin Point -> Join-side .
   op join-rec : Join-seg1 Join-side Join-seg2 -> Join-rec .
   op fixed-seg1 : Join-pt Point -> Fixed-seg1 .
   op fixed-seg2 : Point Point -> Fixed-seg2 .
   op fixed-side : Join-pt Point -> Fixed-side .
   op fixed-rec : Fixed-seg1 Fixed-side Fixed-seg2 -> Fixed-rec .
   op trans-seg1 : Join-rec Join-pt -> Segment [memo] .
   op trans-seg2 : Join-rec Join-pt -> Segment [memo] .
   op trans-side : Join-rec Join-pt -> Segment [memo] .
   op translated-rec : Segment Segment Segment -> Translated-rec .
   op joining-by-s : Join-rec Fixed-rec -> S-Junc .
   op joining-by-c : Join-rec Fixed-rec -> C-Junc .
   op translation : Join-rec Join-pt -> Translated-rec [memo] .
   op is-S-Junc : S-j-seg M-j-seg J-seg
                  S-f-seg M-f-seg F-seg -> S-Junc .
   op is-C-Junc : J-seg M-j-seg C-j-seg
                  F-seg M-f-seg C-f-seg -> C-Junc .
```

In the above, the data sort `Translated-rec` and its associated operators are 'internal' data abstractions which can be, in a sense, hidden from the use of the module. The operators `joining-by-s` and `joining-by-c` are the specifications of the stretching and contracting operations respectively. The equations to implement these and other related operations are provided in Appendix D.2.1. With this module compiled in the OBJ3 database, we can start to produce instances of junctions which represent what we mean by generic outlines. For example,

```
red joining-by-s(join-rec(join-seg1(origin(8.7,8.3),
                                    point(13.5,12.2)),
                          join-side(origin(8.7,8.3),point(6.5,11.3)),
                          join-seg2(point(6.5,11.3),point(11.5,15.3))),
                 fixed-rec(fixed-seg1(join-pt(5,7),point(5,1)),
                           fixed-side(join-pt(5,7),point(2,7)),
                           fixed-seg2(point(2,7),point(2,1)))) .
rewrites: 3228
result S-Junc: is-S-Junc(s-j-seg(s-opp-pt(2.0,9.744),point(7.81,
    11.6)),sm-j-seg(s-opp-pt(2.0,9.744),join-pt(5.0,7.0)),j-seg(join-
    pt(5.0,7.0),point(9.81,10.9)),s-f-seg(s-opp-pt(2.0,9.744),
    point(2.0,1.0)),sm-f-seg(join-pt(5.0,7.0),s-opp-pt(2.0,9.744)),
    f-seg(join-pt(5.0,7.0),point(5.0,1.0)))
```

A world of wall We now specify an object world of 'wall' which contains constructs and operations for expressing more specific information about walls. This includes, for example, 'material' (e.g., brick, block, concrete, etc.), 'type of walls' (e.g., internal and external)[5]. Our current world of wall also includes some special terms given in the design example, namely, types of 'face' and 'end' that constitute a wall junction containing a fixed-wall and a joining-wall. The following shows some parts of the module `WALL`. It makes no claim to specify a 'standard' wall design—it simply illustrates a possible world of describing a wall design.

```
obj WALL is protecting JOINING .
  sorts Material Wall-Type .
  sorts F-face1 End1 End2 F-face2 J-face1 J-face2 .
  sorts Fixed-Wall Join-Wall Junc-Wall .
```

[5]As a simplification, we only give symbolic terms here to denote the information about walls. In theory, it is possible to specify further geometric constructions (some graphic templates, for instance) specific to the detail design of walls.

```
op brick : -> Material .
op block : -> Material .
op concrete : -> Material .
op internal-wall : -> Wall-Type .
op external-wall : -> Wall-Type .
op is-wall-type-of : Wall-Type -> Wall-Type .
op is-material-of : Material -> Material .
op is-SJ-face1 : S-j-seg -> J-face1 .
op is-SF-face1 : S-f-seg -> F-face1 .
op is-End1 : M-j-seg -> End1 .
op is-End2 : M-f-seg -> End2 .
op is-F-face2 : F-seg -> F-face2 .
op is-J-face2 : J-seg -> J-face2 .
op is-J-face1 : J-seg -> J-face1 .
op is-F-face1 : F-seg -> F-face1 .
op is-CF-face2 : C-f-seg -> F-face2 .
op is-CJ-face2 : C-j-seg -> J-face2 .
op is-Fixed-Wall : F-face1 End2 F-face2 -> Fixed-Wall .
op is-Join-Wall : J-face1 End1 J-face2 -> Join-Wall .
op junc-wall : Fixed-Wall Join-Wall -> Junc-Wall .
```

As shown in the above, to make connections between the various types of segments in the module `JOINING` and the various types of face/end in the current module, several predicates are specified (i.e., `is-SJ-face1`, `is-SF-face1` etc.). However, no equations are given here. A full connection between the construction of generic junctions and the description of walls requires further specification of a theory and a parameterised module.

A theory of substantiation In considering how to give the substances of wall to instances of generic junction, we develop a requirement theory of substantiation. The theory is basically built upon an observation of the `WALL` module. It presents a hierarchy of general data types in terms of 'complex', 'objects', and 'components'. The current hierarchical structure in the theory is presented by referring to what was described in the world of wall. The main part of the theory's signature is shown below.

```
th SUBSTANCE is protecting JOINING .
  sort  Complex .
  sorts Object1 Object2 .
  sorts Compnt1 Compnt2 Compnt3 .
  sorts Compnt4 Compnt5 Compnt6 .
  sorts Substance1 Substance2 .
```

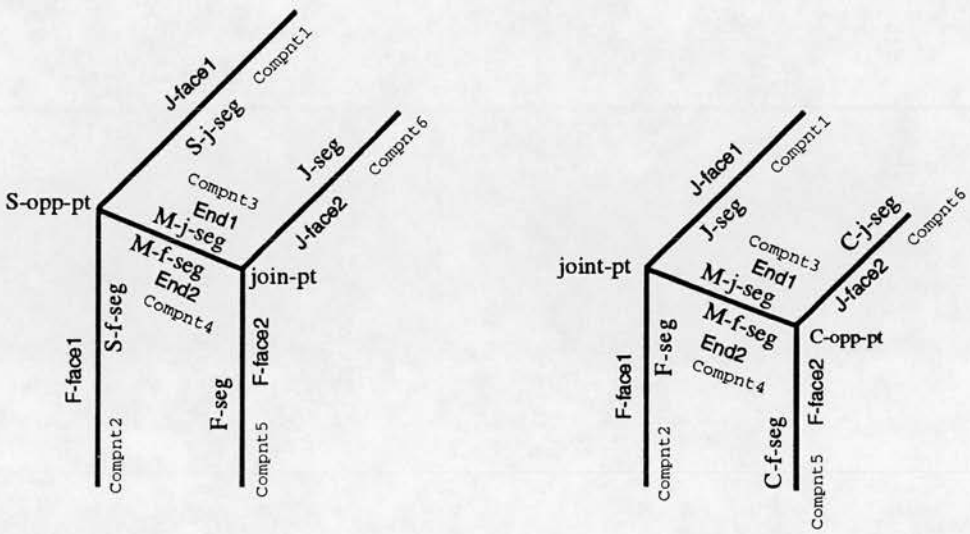


Figure 7.5. A sort mapping scheme from the theory SUBSTANCE to the actual parameter WALL.

```
op declare1 : Substance1 -> Substance1 .
op declare2 : Substance2 -> Substance2 .
op assign1 : S-j-seg -> Compnt1 .
op assign2 : S-f-seg -> Compnt2 .
op assign3 : M-j-seg -> Compnt3 .
op assign4 : M-f-seg -> Compnt4 .
op assign5 : F-seg -> Compnt5 .
op assign6 : J-seg -> Compnt6 .
op assign7 : J-seg -> Compnt1 .
op assign8 : F-seg -> Compnt2 .
op assign9 : C-f-seg -> Compnt5 .
op assign10 : C-j-seg -> Compnt6 .
op comprise1 : Compnt2 Compnt4 Compnt5 -> Object1 .
op comprise2 : Compnt1 Compnt3 Compnt6 -> Object2 .
op constitute : Object1 Object2 -> Complex .
```

Notably, there are a number of 'assigning' predicates presented here, which turn the types of segments defined in JOINING into the types of components defined in the current module. It seems that a scheme of a mapping from the theory to the WALL module is emerging. We shall first present the mapping in the view module next.

A view from SUBSTANCE to WALL Figure 7.5 depicts what a mapping from our theory of substantiation to the WALL module is supposed to do. Following the mapping scheme, the sort mapping in the view WALLV is shown below:

```
view WALLV from SUBSTANCE to WALL is
      sort Substance1 to Wall-Type .
      sort Substance2 to Material .
      sort Compnt1 to J-face1 .
      sort Compnt2 to F-face1 .
      sort Compnt3 to End1 .
      sort Compnt4 to End2 .
      sort Compnt5 to F-face2 .
      sort Compnt6 to J-face2 .
      sort Object1 to Fixed-Wall .
      sort Object2 to Join-Wall .
      sort Complex to Junc-Wall .
```

Given the sort mapping as above, the operator mapping can be followed. For a full presentation of the view, see Appendix D.2.5.

A parameterised substantiating function We now specify what is to be achieved by substantiation for the design of wall junctions. Basically, we want to see types of 'design products' at the end of substantiation. A description of products should contain combined expressions from the initial construction of generic junctions and the wall substances intended to be given. Taking the theory of SUBSTANCE as its interface, we produce the following parameterised substantiation function:

```
obj SUBSTANTIATION[X :: SUBSTANCE] is protecting JOINING .
  sorts Design1 Design2 .
  op substantiate1 : S-Junc Substance1 Substance2 -> Design1 .
  op substantiate2 : C-Junc Substance1 Substance2 -> Design2 .
  op isDesign1 : Substance1 Substance2 Complex -> Design1 .
  op isDesign2 : Substance1 Substance2 Complex -> Design2 .
  vars F1 F2 : Float .
  var S1 : Substance1 . var S2 : Substance2 .
  var SJUNC : S-Junc . var CJUNC : C-Junc .
  eq substantiate1(SJUNC,S1,S2) = isDesign1(
                  declare1(S1),declare2(S2),constitute(
                  comprise1(assign2(s4-S-Junc(SJUNC)),
                            assign4(s5-S-Junc(SJUNC)),
```

```
                        assign5(s6-S-Junc(SJUNC))),
                comprise2(assign1(s1-S-Junc(SJUNC)),
                        assign3(s2-S-Junc(SJUNC)),
                        assign6(s3-S-Junc(SJUNC)))))  .

  eq substantiate2(CJUNC,S1,S2) = isDesign2(
                declare1(S1),declare2(S2),constitute(
                comprise1(assign8(s4-C-Junc(CJUNC)),
                        assign4(s5-C-Junc(CJUNC)),
                        assign9(s6-C-Junc(CJUNC))),
                comprise2(assign7(s1-C-Junc(CJUNC)),
                        assign3(s2-C-Junc(CJUNC)),
                        assign10(s3-C-Junc(CJUNC))))) .
endo
```

The two constructors `substantiate1(2)` take the types of junction in `JOINING` and the types of substance defined in the theory and return the type of design products. How design products are described is implemented in the specifications of the two predicate operators `isDesign1(2)`. Since the mapping scheme is already given in the view `WALLV`, we can decide easily which assigning operator will deal with which type of segment in the junction in formulating the equations.

An instantiated substantiation module What we have in our parameterised specification are the theory `SUBSTANCE`, the parameterised module `SUBSTANTIATION[X :: SUBSTANCE]`, the actual parameter `WALL`, and the view `WALLV`. When all these modules are compiled in the OBJ3 database, we give the system the following module expression for instantiation: `make JUNC-WALL is SUBSTANTIATION[WALLV] endm`

Reduction in `JUNC-WALL` As a result of the above instantiation, a new module, named as `JUNC-WALL`, is generated. Within the instantiated module, a generic junction outline modelled in `JOINING` can be substantiated with some wall design elements into an expression of wall junction. The following is an example of running the module:

```
red substantiate1(is-S-Junc(s-j-seg(s-opp-pt(2.0,9.744),point(7.81,
    11.6)),sm-j-seg(s-opp-pt(2.0,9.744),join-pt(5.0,7.0)),j-seg(join-
    pt(5.0,7.0),point(9.81,10.9)),s-f-seg(s-opp-pt(2.0,9.744),
    point(2.0,1.0)),sm-f-seg(join-pt(5.0,7.0),s-opp-pt(2.0,9.744)),
    f-seg(join-pt(5.0,7.0),point(5.0,1.0))),internal-wall,brick) .
rewrites: 7
result Design1: isDesign1(is-wall-type-of(internal-wall),
```

```
is-material-of(brick),junc-wall(is-Fixed-Wall(is-SF-face1(
s-f-seg(s-opp-pt(2.0,9.744),point(2.0,1.0))),is-End2(sm-f-
seg(join-pt(5.0,7.0),s-opp-pt(2.0,9.744))),is-F-face2(f-seg(join-
pt(5.0,7.0),point(5.0,1.0)))),is-Join-Wall(is-SJ-face1(s-j-seg(s-
opp-pt(2.0,9.744),point(7.81,11.6))),is-End1(sm-j-seg(s-opp-pt(
2.0,9.744),join-pt(5.0,7.0))),is-J-face2(j-seg(join-pt(5.0,7.0),
point(9.81,10.9))))))
```

With the above seven blocks of OBJ3 specifications, we believe that we have presented a symbolic simulation of the design substantiation described in the first design example. This is a simulation of a case in which a single designer carries out the design task. Our second design example depicts a case of joint substantiation in which two designers participate. In the next subsection, we shall continue to present an OBJ3 simulation of the joint case as an extension of the current one.

7.4.2 A simulation of joint design substantiation

In the second design example, the design of a wall junction is taken part in by two designers, Designer A and B. To simulate what is involved in the joint case, we need to specify another series of OBJ3 modules on the basis of the previous simulation. These include the derivative operations for extracting parts of a generic junction as derivative structures, and two individual design modules, with which Designer A and B can perform domain-specific substantiation of partial outlines derived from the same junction outline. Therefore, a major part of the simulation requires two separate instantiations of a (common) parameterised function with respect to the two designers' own actual object worlds and views.

Derivative operations Suppose that Designer A is in charge of the part of a junction on the side of a fixed-rectangle; while Designer B is in charge of the joining side. A and B need to get access to derivative operations for extracting the part of a junction that particularly concerns them. For this, a number of operations are specified in the module `DERIVING` below, which is in fact an extension of the previous `JOINING` module:

```
obj DERIVING is protecting JOINING .
  sorts SJ-DS1 SJ-DS2 CJ-DS1 CJ-DS2 .
  op derive1 : S-Junc -> SJ-DS1 . ***Junction by stretching
  op derive2 : S-Junc -> SJ-DS2 .
  op derive3 : C-Junc -> CJ-DS1 . ***Junction by contracting
  op derive4 : C-Junc -> CJ-DS2 .
```

```
op isSJ-DS2 : S-j-seg M-j-seg J-seg -> SJ-DS2 .
op isSJ-DS1 : S-f-seg M-f-seg F-seg -> SJ-DS1 .
op isCJ-DS2 : J-seg M-j-seg C-j-seg -> CJ-DS2 .
op isCJ-DS1 : F-seg M-f-seg C-f-seg -> CJ-DS1 .
var SJUNC : S-Junc . var CJUNC : C-Junc .
eq derive2(SJUNC) = isSJ-DS2(s1-S-Junc(SJUNC),s2-S-Junc(SJUNC),
                             s3-S-Junc(SJUNC)) .
eq derive1(SJUNC) = isSJ-DS1(s4-S-Junc(SJUNC),s5-S-Junc(SJUNC),
                             s6-S-Junc(SJUNC)) .
eq derive4(CJUNC) = isCJ-DS2(s1-C-Junc(CJUNC),s2-C-Junc(CJUNC),
                             s3-C-Junc(CJUNC)) .
eq derive3(CJUNC) = isCJ-DS1(s4-C-Junc(CJUNC),s5-C-Junc(CJUNC),
                             s6-C-Junc(CJUNC)) .
```

The operators `derive1` and `derive3` are used by Designer A for selecting the data types `SJ-DS1` (the derived structure from the fixed side of a junction formed by stretching) and `CJ-DS1` (the derived structure from the joining side of a junction formed by contraction). Similarly, the operators **`derive2`** and **`derive4`** are available for Designer B to use.

A theory of substantiation Since we assume that there are no new elements appearing in the construction of a generic junction here, the same theory of substantiation specified in the previous example (see Appendix D.2.3) can still be employed in the present case.

A parameterised function for joint substantiation For Designer A and B, the ultimate goal of collaborative design substantiation is to produce domain design expressions in their own individual design modules. Since the intakes of substantiation functions have changed from generic junctions to derivative ones, the parameterised module in the previous example has to be expanded to accommodate the changes. The following parameterised module is provided for both Designer A and B for later instantiations:

```
obj JOINT-SUBSTAN[X :: SUBSTANCE] is protecting DERIVING .
sorts  Domain-A-Exp Domain-B-Exp .
op substantiate-a1 : SJ-DS1 Substance1 Substance2 ->
                                 Domain-A-Exp .
op substantiate-b1 : SJ-DS2 Substance1 Substance2 ->
                                 Domain-B-Exp .
op substantiate-a2 : CJ-DS1 Substance1 Substance2 ->
```

```
                                    Domain-A-Exp .
op substantiate-b2 : CJ-DS2 Substance1 Substance2 ->
                                    Domain-B-Exp .
op isDDEa : Substance1 Substance2 Object1 -> Domain-A-Exp .
op isDDEb : Substance1 Substance2 Object2 -> Domain-B-Exp .
var S1 : Substance1 . var S2 : Substance2 .
var SJDS1 : SJ-DS1 . var SJDS2 : SJ-DS2 .
var CJDS1 : CJ-DS1 . var CJDS2 : CJ-DS2 .
eq substantiate-b1(SJDS2,S1,S2) = isDDEb(
                    declare1(S1),declare2(S2),
                    comprise2(assign1(s-S-j-seg(SJDS2)),
                              assign3(s-M-j-seg(SJDS2)),
                              assign6(s-J-seg(SJDS2)))) .
eq substantiate-a1(SJDS1,S1,S2) = isDDEa(
                    declare1(S1),declare2(S2),
                    comprise1(assign2(s-S-f-seg(SJDS1)),
                              assign4(s-M-f-seg(SJDS1)),
                              assign5(s-F-seg(SJDS1)))) .
eq substantiate-b2(CJDS2,S1,S2) = isDDEb(
                  declare1(S1),declare2(S2),
                  comprise2(assign7(s-J-seg(CJDS2)),
                            assign3(s-M-j-seg(CJDS2)),
                            assign10(s-C-j-seg(CJDS2)))) .
eq substantiate-a2(CJDS1,S1,S2) = isDDEa(
                  declare1(S1),declare2(S2),
                  comprise1(assign8(s-F-seg(CJDS1)),
                            assign4(s-M-f-seg(CJDS1)),
                            assign9(s-C-f-seg(CJDS1)))) .
endo
```

As we can see, the operators `substantiate-a1` & `-a2` are set up for producing substantiated design expressions in A's domain, and the operators `substantiate-b1` & `-b2` are targeted at B's design domain.

Designer A's world of wall-making Suppose that Designer A purports a particular body of knowledge about wall design, covering, say, the kinds of material for building a wall, and the instalment of various types of windows on a wall. In representing his domain knowledge in the current working context, he will need to incorporate the types of constructs that appear in the derivative structures under his modelling. With his world of wall-making, Designer A may proceed to substantiate some part of a generic junction design into a more specific wall design (say, Wall-A, to associate the design authorship with the design product). The following module is a simulation of

A's knowledge about wall design:

```
obj WALL-A is protecting JOINING .
  sort Hide .
  sort Wall-A .
  sorts Material Install .
  sorts F-face1 End2 F-face2 .
  op brick : -> Material .   op block : -> Material .
  op concrete : -> Material .   op window-1 : -> Install .
  op window-2 : -> Install .   op window-3 : -> Install .
  op is-SF-face1 : S-f-seg -> F-face1 .
  op is-End2 : M-f-seg -> End2 .
  op is-F-face2 : F-seg -> F-face2 .
  op is-F-face1 : F-seg -> F-face1 .
  op is-CF-face2 : C-f-seg -> F-face2 .
  op is-Wall-A : F-face1 End2 F-face2 -> Wall-A .
  op install : Install -> Install .
  op is-material-of : Material -> Material .
  op hide : Hide -> Hide .
```

The data type `Hide` and the operator `hide` are devised here purely for the convenience of theory interpretation. As OBJ3 requires a complete mapping of all the sorts and operators specified in a theory, a 'token-like' data type and operator is provided in the target module. This device will be used again in simulating Designer B's individual object world.

Designer A's view Having specified the module `WALL-A`, Designer A may proceed to produce a view which gives an interpretation of how the theory `SUBSTANCE` may be mapped onto his actual module. As the current theory covers a greater ground than the target module `WALL-A`, the sorts and operators appearing in the theory but bearing no correspondence to the target have to be 'masked' by the mapping of to `Hide` and to`hide`. The view `VIEW-A` shown below represents Designer A's mapping scheme:

```
view VIEW-A from SUBSTANCE to WALL-A is
  sort Complex to Hide .    sort Object2 to Hide .
  sort Object1 to Wall-A .
  sort Compnt2 to F-face1 .
  sort Compnt4 to End2 .
  sort Compnt5 to F-face2 .
  sort Compnt1 to Hide .   sort Compnt3 to Hide .
```

```
  sort Compnt6 to Hide .
  sort Substance1 to Material .
  sort Substance2 to Install .
  op declare1 to is-material-of .
  op declare2 to install .
  op assign2 to is-SF-face1 .
  op assign4 to is-End2 .
  op assign5 to is-F-face2 .
  op assign9 to is-CF-face2 .
  op assign8 to is-F-face1 .
  op assign1 to hide . op assign3 to hide .
  op assign6 to hide . op assign7 to hide .
  op assign10 to hide .
  op comprise1 to is-Wall-A .
  op comprise2 to hide . op constitute to hide .
```

Designer A's instantiation From Designer A's viewpoint, he is now in a position to instantiate a new module which can support design substantiation in the domain of Wall-A. Designer A's instantiation involves the theory `SUBSTANCE`, the parameterised module `JOIN-SUBSTAN`, his personal view specified in `VIEW-A`, and the following module expression of make: `make WALL-A-OF-JUNC is JOINT-SUBSTAN[VIEW-A] endm`

Designer B's world of wall-making Similar to the above simulation of Designer A's domain of design substantiation, the following actual parameter and view are specified to simulate Designer B's domain.

```
obj WALL-B is protecting JOINING .
  sort Hide .
  sort Wall-B .
  sorts Cladding Fenestrate .
  sorts J-face1 End1 J-face2 .
  op ext-cladding : -> Cladding .
  op int-cladding : -> Cladding .
  op window-a-door-1-window-b : -> Fenestrate .
  op window-c-door-2-window-d : -> Fenestrate .
  op is-SJ-face1 : S-j-seg -> J-face1 .
  op is-End1 : M-j-seg -> End1 .
  op is-J-face2 : J-seg -> J-face2 .
  op is-J-face1 : J-seg -> J-face1 .
  op is-CJ-face2 : C-j-seg -> J-face2 .
  op is-Wall-B : J-face1 End1 J-face2 -> Wall-B .
  op is-cladded-with : Cladding -> Cladding .
```

```
  op fenestrate : Fenestrate -> Fenestrate .
  op hide : Hide -> Hide .
```

Designer B's view

```
view VIEW-B from SUBSTANCE to WALL-B is
  sort Complex to Hide . sort Object1 to Hide .
  sort Object2 to Wall-B .
  sort Compnt1 to J-face1 .
  sort Compnt3 to End1 .
  sort Compnt6 to J-face2 .
  sort Compnt2 to Hide . sort Compnt4 to Hide .
  sort Compnt5 to Hide .
  sort Substance1 to Cladding .
  sort Substance2 to Fenestrate .
  op declare1 to is-cladded-with .
  op declare2 to fenestrate .
  op assign1 to is-SJ-face1 .
  op assign3 to is-End1 .
  op assign6 to is-J-face2 .
  op assign7 to is-J-face1 .
  op assign10 to is-CJ-face2 .
  op assign2 to hide . op assign4 to hide .
  op assign5 to hide . op assign8 to hide .
  op assign9 to hide .
  op comprise2 to is-Wall-B .
  op comprise1 to hide . op constitute to hide .
```

Designer B's instantiation Having compiled the theory, the parameterised module, his actual parameter `WALL-B`, and his personal view `VIEW-B`, Designer B also reaches the position to instantiate a new module by typing in the following expression:
`make WALL-B-OF-JUNC is JOINT-SUBSTAN[VIEW-B] endm`

As a result, the new module `WALL-B-OF-JUNC` is generated, in which B can carry out domain design substantiation.

Reductions in domain design modules In the above, we have demonstrated how the two designers may set up their individual modelling spaces for taking part in joint substantiation of common generic structures. In the following, we shall see some instances of collaborative design by running the OBJ3 modules.

First, Design A and B jointly construct an initial configuration in terms of an expression of `JOIN-REC` and of `FIXED-REC`. When the operation `joining-by-s` is applied

onto the configuration, a junction of S-Junc is formed, which serves both A and B as a shared generic outline.

```
red in JOINING : joining-by-s(join-rec(join-
seg1(origin(8.7,8.3),point(13.5,12.2)),
                          join-side(origin(8.7,8.3),point(6.5,11.3)),
                          join-seg2(point(6.5,11.3),point(11.5,15.3))),
                fixed-rec(fixed-seg1(join-pt(5,7),point(5,1)),
                          fixed-side(join-pt(5,7),point(2,7)),
                          fixed-seg2(point(2,7),point(2,1)))) .
rewrites: 384
result S-Junc: is-S-Junc(s-j-seg(s-opp-pt(2.0,9.744),point(7.81,
    11.6)),sm-j-seg(s-opp-pt(2.0,9.744),join-pt(5.0,7.0)),j-seg(join-
    pt(5.0,7.0),point(9.81,10.9)),s-f-seg(s-opp-pt(2.0,9.744),
    point(2.0,1.0)),sm-f-seg(join-pt(5.0,7.0),s-opp-pt(2.0,9.744)),
    f-seg(join-pt(5.0,7.0),point(5.0,1.0)))
```

Given the junction outline shaped as above, Designer A applies the derivative operation derive1 so that a derivative structure of SJ-DS1 can be extracted. By adding the wall material and a window template to the extracted outline, Designer A further applies the substantiating function substantiate-a1 and gets a design expression in A's domain. This working procedure can be simulated in the following OBJ3 reduction:

```
reduce in WALL-A-OF-JUNC : substantiate-a1(derive1(is-S-Junc(s-
     j-seg(s-opp-pt(2.0,9.744),point(7.81,11.6)),sm-j-seg(s-opp-
     pt(2.0,9.744), join-pt(5.0,7.0)),j-seg(join-pt(5.0,7.0),point(9.81,
    10.9)),s-f-seg(s-opp-pt(2.0,9.744),point(2.0,1.0)),sm-f-seg(join-
     pt(5.0,7.0),s-opp-pt(2.0,9.744)),f-seg(join-pt(5.0,7.0),point(5.0,
    1.0)))),brick,window-2)
rewrites: 8
result Domain-A-Exp: isDDEa(is-material-of(brick),install(window-
    2),is-Fixed-Wall(is-SF-face1(s-f-seg(s-opp-pt(2.0,9.744),
    point(2.0,1.0))),is-End2(sm-f-seg(join-pt(5.0,7.0),s-opp-
    pt(2.0,9.744))),is-F-face2(f-seg(join-pt(5.0,7.0),point(5.0,1.0)))))
```

In parallel to A's work, Designer B can carry out his part of design substantiation in his own way. By extracting a derivative structure of SJ-DS2 from the same generic junction outline, B is able to produce his domain design expression, containing the information about a template of internal cladding, a fenestration pattern, and a description of Wall-B. The following reduction is to simulate B's design work:

```
red in WALL-B-OF-JUNC : substantiate-b1(derive2(is-S-Junc(s-j-
        seg(s-opp-pt(2.0,9.744),point(7.81,11.6)),sm-j-seg(s-opp-
        pt(2.0,9.744),join-pt(5.0,7.0)),j-seg(join-pt(5.0,7.0),
        point(9.81,10.9)),s-f-seg(s-opp-pt(2.0,9.744),point(2.0,1.0)),
        sm- f-seg(join-pt(5.0,7.0),s-opp-pt(2.0,9.744)),
        f-seg(join-pt(5.0,7.0),point(5.0,1.0)))),
        int-cladding,window-c-door-2-window-d) .
rewrites: 8
result Domain-B-Exp: isDDEb(is-cladded-with(int-cladding),
        fenestrate(window-c-door-2-window-d),is-Wall-B(is-SJ-
        face1(s-j-seg(s-opp-pt(2.0,9.744),point(7.81,11.6))),
        is-End1(sm-j-seg(s-opp-pt(2.0,9.744), join-pt(5.0,7.0))),is-J-
        face2(j-seg(join-pt(5.0,7.0),point(9.81,10.9)))))
```

7.5 Discussion

Design substantiation as a basic form of design modelling is a concept derived from our understanding of the structuralist approach to teamwork in design. Basically, detail design is developed from iterative substantiation of a generic or abstract structure established in the first place. This concept can be applied to individual as well as to group design practice. The purpose of carrying out the OBJ3 simulation in this chapter is to have a closer look at design substantiation in computing terms. We set up the design scenario by introducing an example of wall design. A general and a joint case are described, which provide the work contents to be simulated.

In the general case, a generic structure of a wall design is built upon a set of controlling construction lines and associated transformation rules. As shown in our OBJ3 modelling, a set of constructs and operations, bearing no obvious connection with any elements of a wall as commonly perceived, are coded for the production of outline designs. Design substantiation in this case is the adding of wall-related substances to an instance of an outline design. The parameterisation and instantiation of OBJ3 is employed to simulate what is involved in substantiation. As a result, design substantiation can be performed in a modelling space which is generated by instantiating a parameterised module (defining substantiation functions) with a particular view (defining a mapping from a theory to an actual parameter). By applying the substantiation function to a design expression, we then demonstrate a specific example where an outline design is combined with descriptions of substance and results in a wall junction design.

As an extension to the general case, a case of joint design substantiation is also simulated. The wall junction design is participated by two designers who represent

their expertise in two different worlds of wall-making. As shown in our OBJ3 code, the designers share, firstly, a common modelling space where an outline design can be constructed. By applying the derivative operations defined in the common modelling space, the designers can derive parts of the shared outline design for substantiation in their individual design worlds. The simulation shows that the participants also share a parameterised module that defines a set of substantiation functions. Within this formal framework, design collaboration is based on participants' setting up individual modelling spaces by instantiating the common parameterised module with individual design views. Given the case thus modelled in OBJ3, we demonstrate an example showing that parts of an initial generic outline design can be substantiated into parts of a wall junction design in a distributed manner.

We do not claim that our OBJ3 symbolic simulation presented above has exhausted all the aspects of the structuralist approach to collaborative design. For instance, the constraining forces that may act upon parts of a generic structure, and the interaction between domain-oriented design developments and the state of a common generic structure are not covered in the present simulation. Nevertheless, given the simulation thus far, we may proceed to draw some implications for a system development strategy that considers joint substantiation of common generic structures to be a potential form of collaborative design computing. Let us look first at the following 'collaboration tasks' as prompted by our present simulation:

- *Construction of generic outlines* — This may not have been shown clearly in the simulation. But we may think of the module `JOINING` as being a common ground for construction brought up by Designer A and B jointly, each of whom invents or designates some general constructs for building up common generic design outlines. We have used the term 'initial configuration' to denote the provision of source expressions at the very beginning stage. As a trivial illustration, this is as though Designer A constructs a fixed-rectangle, B a joining-rectangle; and then the two source expressions are put together in a form recognisable to the stretching or contraction operation.

- *Coordination in changing generics* — The reduction results shown earlier can be used to illustrate coordination between Designer A and B in making design changes. The proposition is that, on seeing the domain design expression, Designer A or B may give their judgement on the resultant design from a domain-oriented point of view. Suppose that B is not totally satisfied with his design

result, and subsequently decides to make a change. How will B effect his intended change?

Seen in our simulation, we may say that B has to go back to the initial configuration, which was proposed jointly with A in the first instance, and manipulate parts of it so as to produce a new state of junction outline. Given the new outline, and thus a new derivative structure, B may be able to achieve a more satisfactory domain design expression subsequently. However, since Designer A's domain design expression always contains parts of the junction outline, B's change means that A can no longer maintain the current status of his expression. Communication and coordination is needed if Designer A finds that a re-substantiation of the new outline causes further problems from his own point of view.

Now considering how to facilitate the above collaboration tasks, what would be the system features of a collaborative design environment? In general, the environment should be a network of a group modelling space (GMS) and a number of individual modelling spaces (IMSs). More specifically, we would suggest the following aspects to be developed into potential system components:

1. *Setting up a GMS* — A GMS is a common workspace in which generic structures are constructed and manipulated by members of a design team. According to our simulation, there are a number of components to be established in a GMS:

   - *Basic constructs.* The original status of a GMS given to a design team may be simply a geometric construction space, providing a (minimal) set of geometric constructs such as construction lines and points. On top of these initial elements, designers can define whatever 'basic constructs' they like. However, the set of basic constructs should remain as general as possible so that all participants can get access to and manipulate the instances of the basic constructs without particular knowledge.
   - *Derivative operations.* As shown in the `DERIVE` module, derivative functions are defined with respect to how a generic structure can be divided into sub-structures. This implies that derivative operations need to be 'user-definable' at a time when the users come up with their own decisions.

   The above two components seem to suggest an 'interactive specification system' with which designers can interact in producing the constructs and operations they intend to use.

2. *Instantiating IMSs* — Another challenging issue related to the setting up of a GMS is the design of a substantiation theory and a parameterised substantiation function, with which each participant can instantiate his or her IMS. We have given examples of such a theory and function in our simulation. But this is done in a manner that assumes a lot of knowledge about the object world of wall. In reality, this can be impractical. However, we feel that this is a potential area for basic research into a more fundamental theory of design substantiation and its presentation. Only if such a basic theory is sought, may we be in a position to design the facility to assist users' theory interpretation for achieving meaningful instantiation of IMSs.

3. *Bookkeeping design change* — This is an area where we consider that computer-based support can take a more active role. A bookkeeping mechanism can be developed to inform members of a design group about the status of sharing common generic structures. As discussed before, a participant's intention to change parts of a generic structure may potentially cause problems to other designers. The usefulness of a computer-based bookkeeper would be the evaluation and delivery of the consequences of making changes proposed by a designer to other designers' workspaces. A simple outline of designing a bookkeeper of this nature may include the following:

   - Detection or recognition of design changes proposed by some individual in the current state of an initial configuration;
   - Application of form-giving operations onto the changed configuration, resulting in a (pending) state of a generic outline;
   - Applications of derivative operations that have been applied by participants before onto the (pending) state of a generic outline, resulting in, perhaps, a number of (pending) states of derivative structures;
   - Applications of domain-oriented design substantiation functions that have been applied before onto the (pending) states of derivative structures, yielding (pending) states of domain design expressions;
   - Delivery of the generated (pending) design expressions to the right workspaces, alerting the individuals about the consequences of design changes as proposed by some designer(s).

Summary

The sharing of common generic structures is one of the most important features observed in the structuralist approach to collaborative design. Members of a design team have two roles to play: one is to take part in the construction of a generic structure, the other is to substantiate parts of the common structures with domain-specific design expressions. This chapter gives a symbolic simulation of the relations between the two roles. We use a borrowed example to illustrate that design in general can be considered as a substantiation process—the development from a generic outline into a more specific design specification. The structuralist pattern can be seen as an extension of the general (individual) practice to a group practice. To gain a clearer understanding of the relation between the sharing of $\mathcal{CGS}$ and group substantiation of $\mathcal{CGS}$, we have carried out an OBJ3 simulation. With the formal platform provided by OBJ3, we use the scheme of *requirement theories*, *parameterised modules*, and *views* to simulate a case of joint substantiation. The current simulation shows that parameterisation and instantiation may provide a formal basis for developing collaborative design computing that supports the structuralist requirements.

Chapter 8

Conclusions and Further Research

8.1 Supporting Teamwork in Design as Explored

The primary concern of this thesis is to enquire into the requirements for computing systems that can be supportive of human collaborative design activity. Our enquiry starts by asking "what do designers' design expressions, as seen in our case studies, reveal about group dynamics in building design?". Considering the nature of design in relation to what has been experienced in computer-aided design, we propose the perspective of design as an activity of modelling complex objects. From this perspective, we set out to elucidate the requirements for computer support by examining what constitutes *teamwork in architectural modelling.* In particular, we think that it would be fruitful if a better understanding and, eventually, a systematic description of the interrelations between the participation of multiple design expertise and the emergence of final unity in design products can be achieved.

In retrospect, there are four research tasks pursued and reported in this thesis, each of which, in our view, makes a contribution to the requirements study in computer-supported collaborative design:

Case studies and a conceptualisation of teamwork patterns — Three historical cases of building projects are studied. In each case, there is a collection of design expressions produced by the members of the design groups. In our view, these expressions represent, to a certain extent, how design is distributed and integrated. The case observations have led us to a conceptualisation of two distinct

teamwork patterns in collaborative design: metaphorist and structuralist. Admittedly, from so limited a study, these patterns are not claimed to be definitive or exhaustive, but provide a useful starting point for analysis. For these teamwork patterns identified, we subsequently develop more elaborate accounts of the representation and communication requirements.

A survey of collaborative drawing support tools — To be aware of the related work undertaken by other researchers in the field of computer-supported cooperative work, we conducted a survey of the recent experimentation in supporting collaborative drawing and design activities. Various prototype systems have been developed in the past few years from much diversified perspectives of what to facilitate in collaborative drawing and design. There are systems developed as a shared visual and action space for a group of designers to undertake computer- and/or video-augmented (face-to-face) design meetings. Making use of modern networking technology and distributed systems, some system researchers have shown the potentials of supporting simultaneous group drawing activity participated in by designers working remotely in separate workspaces. Being developed from an artificial intelligence standpoint, some systems are built upon a representation of the structure of some organisational procedures, design tasks or artefacts targeted at specific application domains.

Though recent CSCW research has shown some interesting results concerning how communication and computing tools can be devised to facilitate group design work, there are research areas unexplored by the current technological solutions. Alternative system strategies may be suggested if we can have a fresh understanding of the requirements for supporting collaborative design.

Analyses and descriptions of the requirements for computer support — To better describe what is involved in collaborative design, we adopt a situation-theoretical framework in our further account of the metaphorist and structuralist teamwork patterns. In these 'quasi-formal' analyses, each teamwork pattern is interpreted as a pattern of information flow. Designers are said to work with one another in a network consisting of a shared group modelling space and, perhaps, several distributed individual modelling spaces.

Participants' performing various modelling acts in the network give rise to the flow of design information. Given the different types of information carriers (i.e., types of design representation) classified, the differences between the metaphorist and the structuralist approach are presented more sharply in terms

of the directions of information flow analysed. For design information to flow from one type of design representation to another, certain constraints need to be satisfied. It is by clarifying the constraints that we arrive at the representation and communication requirements for each teamwork pattern.

Simulation of two potential forms of collaborative design computing — To explore how our conceptual understandings of teamwork in design can be related to potential forms of collaborative design computing, two computational simulations are carried out. One is a simulation of joint abstraction and use of shared integration schemas in the metaphorist approach, the other, a simulation of joint substantiation of common generic structures in the structuralist approach.

For the purpose of illustration, simple design exercises are described, indicating, mainly, some of the representation aspects in each approach to be simulated. In both simulations, examples of joint abstraction and substantiation are given explicit representations, from which we generate instances of design expressions that illustrate the need for group communication and coordination in making design changes. Given the data acquired through the symbolic simulations, we draw up some implications for system development in response to the metaphorist and structuralist requirements. Seeing design as a modelling activity, human designers will take a more active role in designing with computers. This computer-aided design paradigm gives even a greater demand for system to support teamwork in design. We hence would argue that "collaborative design computing" is an alternative to further the current development of computer-supported collaborative drawing.

Having described what research tasks have been carried out in the thesis, it is considered important to acknowledge the limitations of the current study. First, as mentioned before, the range of case studies is limited. Our conceptualisations of the metaphorist and structuralist teamwork patterns are two among, possibly, many others. Quite likely, different sets of requirements will arise if further patterns are identified in a broadened base of case studies.

Secondly, our requirements study has been based more on theoretical analyses, as opposed to empirical elicitation. This inevitably leads us to the search for rather formal requirements for system support. Our survey of collaborative drawing support tools shows that the informal requirements derived empirically have clear influences on most prototype system developments. However, our effort can be justified by the initial concern that design is essentially a modelling activity, which is not entirely subject to

empirical observation. Also, details of group design behaviour, while important and highly relevant to interface design, may confuse the conceptual issues of modelling.

The third limitation lies in the scope of our symbolic simulations. We realise that in either the metaphorist or the structuralist approach, *visual images* are important mediums for group communication in collaborative design. What we have demonstrated in the OBJ3 simulations are simplified geometric constructions. Obviously, the issues of multi-user interface and the generation of graphical images have not been taken into account. The algebraic specification language enables us to gain structural clarity but not the *fluidity* and *complexity* often associated with real design images.

8.2 Metaphorist vs. Structuralist

To give an overview of the main findings of the thesis, this section presents an excerpt from our current investigations. Putting the case studies, the conceptual analyses, and the computational simulations jointly, we may summarise our findings in terms of the features of the two teamwork patterns and the various requirements for system support.

8.2.1 Features of the teamwork patterns

One of the major findings of the thesis is the differentiation of the two teamwork patterns termed as metaphorist and structuralist. As presented, an understanding of teamwork pattern can help a basic definition of computer support for human collaboration. But how significant is the distinction made between the metaphorist and the structuralist patterns? In this section, we shall reflect on our current finding of the differences and draw in other findings in some related areas.

Table 8.1 summarises the various features observed in the metaphorist and structuralist approaches to collaborative design. We think that the differences between the two teamwork patterns can be listed in the following aspects:

- *Inception* — This is the starting point of teamwork in design. The metaphorist starts with local design decisions made in individual object worlds set up in individual modelling spaces. Employing a shared set of constructs and connectors, the structuralist starts with the construction of a common generic structure in a group modelling space. It seems reasonable to say that how a teamwork project might be developed is closely related to how it begins. If design teams have 'personalities', it is the inception of a design project that makes one team work differently from another.

	Metaphorist	Structuralist
Inception	Individual Modelling Spaces Individual Object Worlds Local Design Decisions	Group Modelling Space Shared Construction Set Common Generic Structures
Design Unity	Common design metaphors	Common generic structures
Design Development	Domain-crossing design integration	Distributed heterogeneous design substantiation
Design Consequences Revealed	Projection of common images on the basis of integrating multiple local design decisions in a GMS	Generations of domain design expressions in distributed IMSs with reference to a common generic structure
Design Change	Judgements on projected common images giving rise to changes to be made in local design decisions	Judgements on resultant domain design expressions giving rise to changes to be made in a common generic structure

Table 8.1. A list of the differences between the metaphorist and the structuralist teamwork patterns as seen in this thesis.

- *Design development* — Complex building projects often last for a long period of time, involving various stages of design development. Design development in the structuralist pattern takes on the task of substantiating common generic structures with domain-oriented design details. The metaphorist, on the other hand, takes on the task of integrating multiple local design decisions into greater wholes.

- *Design unity* — Architectural design in its nature demands certain kinds of unity or wholeness to be achieved. It may be difficult for us to define objectively what is meant by 'design unity' and how the quality of design unity may be achieved and measured. In fact, any judgement of design unity may have to be left to the designers themselves to decide. However, the presentation of design unity has an important role to play in collaborative design since members of a design team can be motivated to develop individual work by perceiving and interpreting its presence. The metaphorist gains design unity by sharing common design metaphors; while the structuralist does so by sharing common generic structures.

- *Design consequences revealed* — Working as members of a design team, how do

designers know that they have achieved something they want both from an individual and a team point of view? To collaborate with others, each member has to work between the goal of design unity and the goal of design development. It is by making constant connections between the two goals, that the consequences of collaborative design are revealed to each individual. The metaphorist becomes aware of design consequences by projecting common images based on an integration of the local design decisions developed by different individuals. In the structuralist case, design consequences are known to group members when domain design expressions are generated on the basis of substantiation of parts of a shared generic structure.

- *Design change* — Design change is intended whenever design consequences appear unsatisfactory to one or more designers. Following the establishment of interrelations between design unity and domain design developments, one or more designers' making design changes may cause others to enter into negotiation or coordination. For the structuralist, by judging resultant domain design expressions, design change is targeted at parts of a common generic structure, which may lead to a call for collaboration in order that the shareability of the generic structure can be recovered. For the metaphorist, by judging the outcome of projected common images, design change is aimed at local design decisions, which may have further effects upon the formation of the common images.

8.2.2 Requirements for collaborative design computing

Drawing from our information-flow analyses of the teamwork patterns together with the OBJ3 simulations of two potential forms of collaborative design computing, it can be said that requirements study, in short, asks three basic questions:

- What do members of a design team share in the processes of modelling a design artefact?
- How may the shared design images be represented in a common workspace?
- How does the representation of whatever is shared affect the interaction among team members when working in individual workspaces?

Following the above three aspects outlined, Table 8.2 gives an overview of the metaphorist and the structuralist requirements for representation and communication support as explored by this thesis.

	Metaphorist	Structuralist
What's shared?	shared integration schemas for projecting integrated design images	common generic structures for distributed domain design substantiations
How the shared is represented?	• spatial operations • combined source designs • integrated design images	• generic outline designs • parameterised substantiation functions • a field of transforming forces
Areas of representation support	— basic constructs for domain-crossing translation and integration — visualisation of integrated design images	— a basic theory of design substantiation — a constraint system for shaping the state of a common generic structure
How the representation of the shared affects group communication?	the unity emerging from design integration gives rise to individual judgements of domain design developments, which, in turn, results in evolution of the unity achieved	individuals' substantiating parts of a common generic structures gives rise to changes to be made in the generic structure, which, in turn, may cause changes in some domain design dvelopment(s) to follow
Areas of communication support	— communication form for domain-crossing translation and integration — detection of changes in integrated design images — generation and delivery of design change reports	— communication form for putting parts of an initial configuration together — bookkeeping design changes in the state of a common generic structure — alerting consequences of design changes in the common generic structure to the individuals related

Table 8.2. A summary of the metaphorist and the structuralist requirements for system supports as explored in this thesis.

Based on a combination of the teamwork features and the requirements for system supports, Figure 8.1 shows a diagrammatic representation of the focal point of the two system development strategies that attend to the metaphorist and the structuralist approaches to collaborative design respectively.

It should be noted that the contrasting nature of the above findings indicates a spectrum of possibilities in computer-supported collaborative design rather than two specific alternatives. However, more research is needed to investigate whether a unification of the current two strategies can lead to an even more comprehensive and general theory.

8.3 Related Studies in Three other Areas

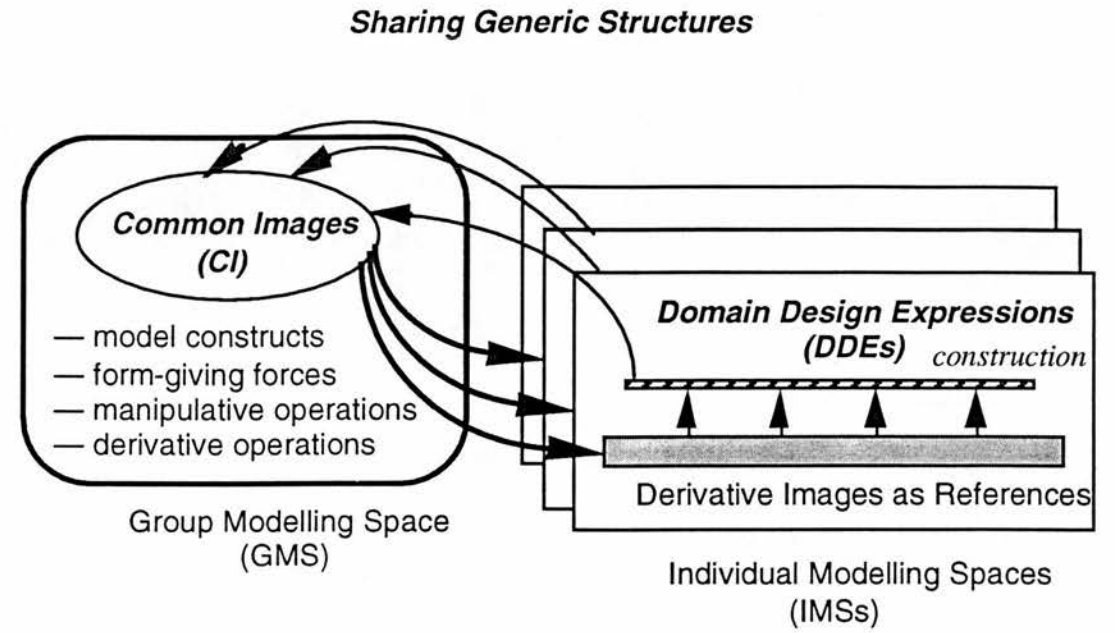
Architectural design has been through a long history. We should not feel surprised to see two, or, indeed, even more, different approaches to the design of architectural buildings as a creative teamwork enterprise. Presumably, any differentiation arising from an enquiry represents an understanding of the subject matter in question. In our case, we would suggest that the design of computer-based support for creative collaborative design should be founded on an understanding of the teamwork patterns in question, attending to the representation and communication requirements to be fulfilled by a system. This thesis presents just two such attempts.

To show that our current understanding of the contrasting features of the teamwork patterns, no matter how limited, is not an isolated finding, we would like to comment on related observations in three other areas: (a) modern debate on architectural design, (b) theory of human knowledge, and (c) enterprise integration (EI) modelling.

8.3.1 Function vs. form in modern architectural debate

Perhaps due to the rapid advancement in modern industrial technologies and the radical changes in social cultural milieu, the development of modern architecture has experienced a sharp split between 'function' and 'form'[1]. Generally speaking, function in architecture and design often refers to the concepts of utilities, services, performance and economy etc. From a functional point of view, design is a means to achieve some practical or utilitarian ends associated with the satisfaction of various human needs

[1]For a scholarly exposition of the modern architectural design debate on form and function, see Banham's seminal work presented in [Ban60]; for a comprehensive collection of modern architectural theorists' and designers' writings on this subject, see the anthology edited by Benton, Benton, and Sharp [BBS75].



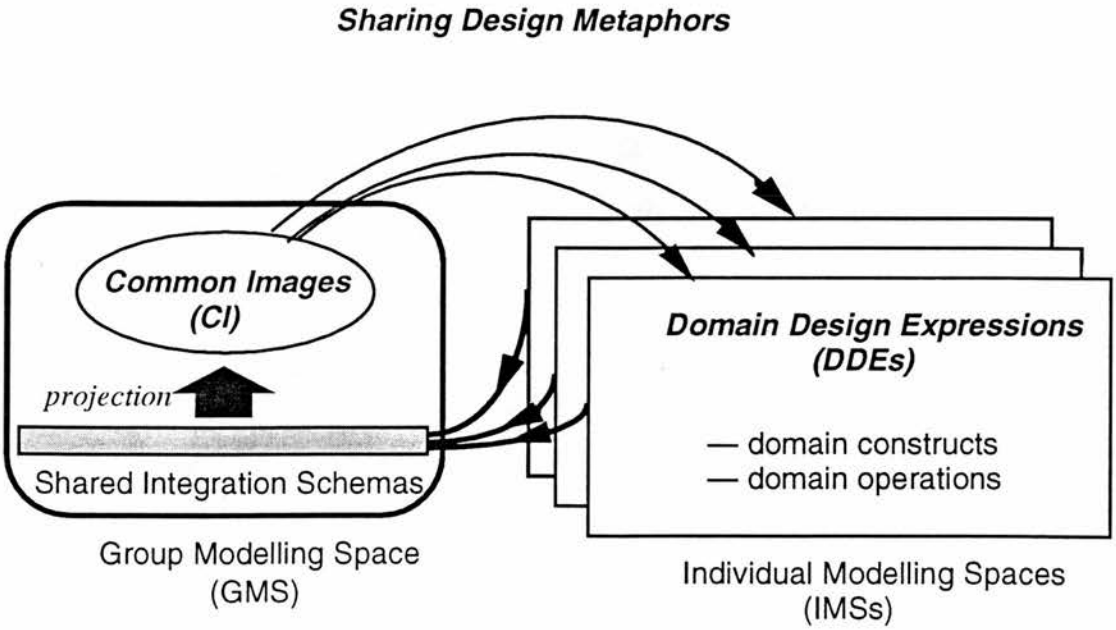


Figure 8.1. A diagrammatic depiction of two system strategies that aim at supporting the metaphorist and the structuralist approaches to collaborative design.

such as biological, ergonomic, economic, social and so on. Form, on the other hand, is more concerned with the ideas of shapes, spaces, mass, and volumes that are often associated with aesthetic judgements and architectural meanings. From the point of view of architectural form, design is an end in itself, which is to serve, mainly, human aesthetic sensitivities.

Given the split between form and function, the modern debate on architectural design (sometimes extended to other design practices such as industrial design, furniture design etc.) centres on the subordination relation between function and form, that is, 'form follows function', or 'function follows form'. This is not a place for us to instigate the debate, but it is interesting to see its relevance to our current study of the approaches to teamwork in architectural modelling.

The metaphorist pattern seems to start with team members' separate concerns with functions and ends up with the projection of a common form[2]. In contrast, the structuralist approach goes for a common form first and, subsequently, develops more specific functions in respect to the parts derived from the common form. However, as we have explored before, in both cases, there is constant interaction between design decisions on form and on function in its overall group design processes.

8.3.2 Collectives and generics in human knowledge

As said before, an understanding of what designers share in the processes of collaborative design is an important aspect of requirements study. What is shared by members of a design team can be considered as a kind of human knowledge. Though it may appear highly contextualised and fluid, a representation of the knowledge can be a useful resource for affecting design communication and coordination. Concerning the nature of the knowledge shared in collaborative design, we find another relevant observation in the area of human knowledge studies (Epistemology).

The English mathematician J. L. Jolley once proposed a theory of human knowledge in which he distinguished two varieties of 'semantic types': the *collectives* and the *generics*. 'Semantic type' is the term Jolley used to denote one of the hierarchical

[2]Arguably, the starting point of the metaphorist approach might be team members' separate concerns with forms deliberated on particular design aspects, which are then interrelated and synthesised into a common form. In dealing with "synthesis of form" in the domain of environmental planning and design, Christopher Alexander has published a formal method [Ale64], with which simple forms can be put together to produce a unified complex form. However, as demonstrated in Alexander's applying the methodology, all the constructions of simple forms (as constituent design parts) are based on (or guided by) the consideration of various functions in the first place.

elements with which he modelled the pattern of human ideas. According to Jolley, semantic types refer to the different types of notions, whose existence has led languages to develop the different parts of speech—nouns, verbs, and adjectives etc.[3] The distinction made between generics and collectives is based on Jolley's further abstraction of the several types of human notions he classifies. To have a general idea of what is meant by generics and collectives in Jolley's terms, the following quote is worth reading [Jol73, p. 37]:

> "Generics are always closely concerned with the construction of the notions they bring together. Iodine is a halogen because of the number of electrons in the outer orbit of each of its atoms. Other elements with the same structure in this respect are also halogens. Iodine is a thing, and so is an electron, and indeed it seems that generic ideas always group ideas of the same semantic type as themselves.
>
> By contrast, a collective idea groups notions which are of a different semantic type from itself. An example is the concept of a weapon. Weapons are things grouped according to a phenomenon, namely their use. Antiques, ironmongery, roofing materials, stock-in-trade, paint, tiles, professors and public relations officers: these, too, are collectives. We live in a world of them."

Jolley's distinction between the generics and the collectives is relevant to our finding of what designers share in collaborative design. As explored in this thesis, to collaborate with one another, designers create and share *common images* ($\mathcal{CI}$); but instances of $\mathcal{CI}$ are of two different natures. In the structuralist approach to teamwork, designers share common generic structures ($\mathcal{CGS}$) that are close to the generic type of human knowledge as characterised by Jolly. The modelling of $\mathcal{CGS}$ is to do with the construction of the notions of an architectural form brought together by a team of designers; and the structuralist construction of $\mathcal{CI}$ is based on a common collection of model constructs. Working in the metaphorist way, designers create and share projected common images that depict integrated design effects or performance. These common images are produced by grouping different design parts together, leading to the visualisation of an overall performance or effect of the designed artefact. As observed, the various design parts are modelled by members of a design team who have discrete concerns of domain-oriented functions. To this, we think that the $\mathcal{CI}$ shared in the metaphorist approach is of the collectives.

As a part of a proposed epistemological pattern, Jolley's differentiation of the collectives and the generics helps to place different notions systematically into a structure

[3]For a more detailed discussion of semantic types in Jolley's terms, see [Jol73, pp. 16–39].

of human knowledge. In investigating how collaborative design may be computer-supported, our concern centres on what is shared by human designers and how the shared affect communication and coordination. As discussed respectively in the previous chapters, whether the design information or knowledge shared is of the generics or of the collectives can give rise to different requirements of how the making of design changes may be supported. In the generic case, it is the change in common generic structures that may cause further changes to follow in distributed domain design developments; but the implications of change in $\mathcal{CGS}$ can only be revealed by looking at new states of domain design developments. In the collective case, on the other hand, it is the change in local design decisions that may cause further change to occur in the projection of common images. The implication of a participant's design change can only be known by communication with other participants who interpret the resultant change in the common image from their own domain viewpoints.

8.3.3 Unification vs. federation in EI modelling

The third related research finding is presented in the area of database system engineering. The concept and implementation of *models* are essential to any serious development of database systems. Sophisticated models and complex relations among the models have been put forward. As we see design as a modelling activity, systems that support collaborative design of this nature will have to resort to methodologies of managing the integration and distribution of design modelling tasks involving multiple individuals.

Researchers working in the field of enterprise integration (EI) modelling have proposed the difference between 'unified' and 'federated' approaches to the issue of *model integration.* According Hars and others [HKLP92], the basic difference is between *early* and *late* binding. In the unified approach, early binding is the case, where applications are 'compiled' from models, that is, all links between models and applications are established by the compiler. If a federated approach is taken, then late binding is the case, where applications and even models are linked only at runtime, as required.

Shown in our OBJ3 simulation of the metaphorist approach, joint abstraction and use of shared integration schemas is potentially a case of unification by early binding, where applications (i.e., the projective or integrative operations) are compiled from individual models presented by participants working in different domains. Collaborative substantiation of common generic structures, as demonstrated in our OBJ3 simulation of the structuralist approach, on the other hand, can be seen as a case of federation by late binding; applications (i.e., domain-oriented design modules) are instantiated

from generic theories, substantiation functions, and various views. Regarding this correspondence, we may say that given our present requirements study accomplished, the computing strategies developed by EI engineers, as briefly mentioned here, and an even wider range of methodologies developed by researchers of database systems, as described by Batini and others in [BLN86], can be a useful technical resource for further development in system design and implementation.

8.4 A Direction for Future Research

In this thesis, we explore patterns of teamwork in architectural modelling; for each teamwork pattern identified, a situation-theoretical analysis of the representation and communication requirements is carried out; and for each potential form of collaborative design computing, as suggested by the requirements analyses, a symbolic simulation is given. On the basis of what we have explored here, the following direction for future research into computer-supported collaborative design is considered worth pursuing:

Emergent communication frameworks. As a general strategy, the way forward for computer-supported collaborative design is somewhere between AI (or Distributed AI) and CSCW. A useful balance to strike is how to integrate formal representations and the design of informal group workspaces to such an effect that the systems can support *emergent communication frameworks* created by participants in the course of designing. More research is needed to investigate how such frameworks may be represented in computers and what functionality is pertinent to support designers' acting upon the 'web' of design communication.

From requirements study to system specification. Up to now, we have used situation theory as a descriptive framework for capturing the information flow perspective on the metaphorist and structuralist scenarios. Our present work is more concerned with descriptions of the representation and communication needs from the human teamwork point of view. To be able to describe precisely what system features can be supportive of these needs, more research into specification of system behaviours is needed. A move from our present requirements study toward a formal system specification requires proper tools. In the field of theoretical computer science, several formal languages have been developed to describe, in particular, the behaviour of concurrency and communication, which is one of the system properties rather hard to specify precisely. Regarding this aspect of further research, the following specification languages are examples of

potential tools for carrying out formal system specification: *Communicating Sequential Processes* (CSP) [Hoa85], *A Calculus of Communicating Systems* (CCS) [Mil89], and *modal* or *temporal logic* as established in [BBP89, Sti92].

Building prototype collaborative design environments. Given the strategy adopted and the formal system specifications constructed, we may arrive at a better position to build up a prototype collaborative design environment for an empirical test. Perhaps, like any other research into system design and development, it is in the testing of the prototype systems that we may come to realise what might have gone wrong, including the prototype mechanisms devised, the system behaviours specified, and the theoretical standpoint adopted.

Appendix A

Glossary

Below are the terms and notations used in this thesis (Chapters 4 and 6, mainly) for describing *constraints on collaborative design* in a shorthand form. Only general readings of these terms and notations are given; a system of formal semantics is *not* intended here.

A.1 A Glossary of the Metaphorist Terms

IMS(s) *Individual Modelling Spaces*: workspaces created and evolved by designers individually for modelling design expressions (e.g., diagrams, drawings, or any other graphical/textual constructions) targeted at a particular design aspect or domain of a design project.

GMS *Group Modelling Space*: a common workspace created and evolved by members of a design team jointly for modelling the integration of design parts contributed by each member into larger design wholes.

$\mathcal{IOW}$ *Individual Object World*: a representation scheme formed by a designer's abstraction act performed (individually) in his or her own IMS.

$\mathcal{LDD}$ *Local Design Decisions*: instances of $\mathcal{LDD}$ are design expressions, showing, for example, what spatial forms or particular functions are modelled by the designer when embarking on domain-specific design tasks. We use $\mathcal{LDD}_a$ to denote a local design decision made by a designer named a.

$\mathcal{SIS}$ *Shared Integration Schema*: design concepts or methods that are developed jointly by members of a design team for the purpose of performing the integration of individual contributions into design wholes. A collection of such shared concepts or methods can be

correlated into an 'integration schema' which can be constantly employed by the design team to handle design integration at a larger scale.

$\mathcal{CI}$ *Common Images*: common images are pictorial representations that are generated by design members' putting proposed $\mathcal{LDD}$s together and then transforming them into single compositions under the operation of shared integration concepts and methods.

$\mathcal{CDM}$ *Common Design Metaphors*: common notions or images reached jointly by members of design group that are taken as common design references to either *what a common image looks like* or *how a common image functions* during design communication.

$\mathcal{DDA}$ *Domain Design Agendas*: an individual's interpretation of a common image may bring out some significance or role of the individual's work in the context of an emerging whole. A domain-specific design agenda is formed in an IMS as a result of an individual's interpretation of $\mathcal{CI}$, which may contain items of information useful in guiding further domain design development.

A.2 A Glossary of the Structuralist Terms

The terms GMS (Group Modelling Space), IMSs (Individual Modelling Spaces), and $\mathcal{IOW}$ (Individual Object World) appearing in the Structuralist pattern share the same readings given in the previous section of the Glossary.

$\mathcal{SCS}$ *Shared Construction Set*: a set of modelling primitives introduced by members of a design team for the modelling of *common generic structures* (see below) in a group modelling space.

$\mathcal{CGS}$ *Common Generic Structures*: common generic structures are 2-D or 3-D objects, representing, mainly, a kind of spatial framework or skeleton that is constructed and can be used by all participants working in different domains of a design project. There are several important properties observed in common generic structures, including *deformability, multi-perspectiveness*, and *genericity*. A more detailed description of these properties of $\mathcal{CGS}$ can be found in Section 6.3.3, page 142.

$\mathcal{DS}$ *Derivative Structures*: derivative structures are 2-D or 3-D objects that are produced by applying derivative operations onto a state of a common generic structure. Images of derivative structures, once imported into individual modelling spaces, can serve the individuals as design referents or outlines in generating more detailed designs of particular design aspects.

$\mathcal{DDE}$ *Domain Design Expressions*: domain design expressions are the outcomes from designers' substantiating derivative structures with design expressions that contain specifications of domain-specific design substances or properties.

$\mathcal{LDJ}$ *Local Design Judgements*: an individual gives an interpretation (or an evaluation) of domain design expressions arrived at an IMS. Since each domain design expression is generated in relation to what derivative structures are underlaid, the resultant $\mathcal{DDE}$, when viewed and judged by its creator from a domain-specific design perspective, is a *design consequence* of a state of a common generic structure.

A.3 Symbols for denoting Constraints on Collaboration

Δ *State Change*: any state change occurring to a type of design state. For instance, $\Delta\mathcal{LDD}_{john}$ refers to a state change in a local design decision made by a Designer named *john*.

$[E_1, E_2, E_3, \cdots]$ *Distributed Entities*: a number of representation entities that are distributed over several individual modelling spaces. For instance, $[\mathcal{LDD}_a, \mathcal{LDD}_b, \cdots]$ refers to a distribution of information entities consisting of a local design decision made by $Designer_a$, a local design decision made by $Designer_b$, and others.

Σ *Collective Presentation or Construction*: states of, for instance, (multiple) local design decisions or changes in local design decisions are collectively presented (assembled) by participants in a common visual space during a design meeting. Σ is also used to denote a resultant common set of constructs developed by collaborative designers.

$\rightsquigarrow$ *Leading to*: a symbol denoting the *flow of information* (in a situation-theoretical sense) from what is contained at the left-hand side of the curled arrow to what is contained at the right-hand side of the arrow.

$\Longrightarrow$ *Involving*: a symbol denoting the *involving* relation between two situation types. Situation theory characterises *constraints* by introducing a primitive relation between types of situations, the relation of *involving*. The involving relation may be introduced on the basis of abstract links that capture (are) systematic regularities (e.g., natural laws, linguistic rules, conventions, logical rules, etc.) connecting situations of one kind with situations of another. With reference to the situation-theoretical framework, a constraint on collaborative design $\mathcal{C}$ can be expressed by a situation, in which there is a *leading to* relation between two different design states, involving another situation $\mathcal{S}'$, by writing

$$Constraint\ \mathcal{C}\ :\ (DesignState\ R_1 \rightsquigarrow DesignState\ R_2) \Longrightarrow \mathcal{S}'$$

Appendix B

Basic Geometric Construction in OBJ3

B.1 A Specification of Angle

```
obj ANGLE is
   protecting FLOAT .
   sorts Radian Degree .
   subsorts Float < Radian .
   subsorts Float < Degree .
   op inradian : Float -> Radian .
   op indegree : Float -> Degree .
   op angle : Float Float Float Float -> Degree .
   op d-to-r : Degree -> Radian .
   op r-to-d : Radian -> Degree .
   vars X1 Y1 X2 Y2 : Float .
   var D : Degree .     var R : Radian .
   cq angle(X1,Y1,X2,Y2) = r-to-d(atan((Y2 - Y1) / (X2 - X1)))
                           if Y2 - Y1 > 0   and   X2 - X1 > 0 .
   cq angle(X1,Y1,X2,Y2) = 180 - r-to-d(atan(abs(Y2 - Y1) / abs(X2 - X1)))
                           if Y2 - Y1 > 0   and   X2 - X1 < 0 .
   cq angle(X1,Y1,X2,Y2) = 180 + r-to-d(atan(abs(Y2 - Y1) / abs(X2 - X1)))
                           if Y2 - Y1 < 0   and   X2 - X1 < 0 .
   cq angle(X1,Y1,X2,Y2) = 360 - r-to-d(atan(abs(Y2 - Y1) / abs(X2 - X1)))
                           if Y2 - Y1 < 0   and   X2 - X1 > 0 .
   cq angle(X1,Y1,X2,Y2) = 0    if Y2 - Y1 == 0 and X2 - X1  > 0 .
   cq angle(X1,Y1,X2,Y2) = 90   if Y2 - Y1  > 0 and X2 - X1 == 0 .
   cq angle(X1,Y1,X2,Y2) = 180  if Y2 - Y1 == 0 and X2 - X1  < 0 .
   cq angle(X1,Y1,X2,Y2) = 270  if Y2 - Y1  < 0 and X2 - X1 == 0 .
   eq d-to-r(D) = pi * (D / 180) .
   eq r-to-d(R) = R * (180 / pi) .
endo
```

B.2 A Specification of Point

```
obj POINT is
    protecting FLOAT .
    sort Point .
    op point : Float Float -> Point .
    ops x y : Point -> Float .
    op distance-pp : Point Point -> Float .
    op _right-to_ : Point Point -> Bool .
    op midpoint : Point Point -> Point .
    vars F1 F2 : Float .    vars P1 P2 : Point .
    eq x(point(F1,F2)) = F1 .
    eq y(point(F1,F2)) = F2 .
    eq distance-pp(P1,P2) = sqrt((y(P2) - y(P1)) * (y(P2) - y(P1)) +
                                 (x(P2) - x(P1)) * (x(P2) - x(P1))) .
    cq P1 right-to P2 = true if x(P1) > x(P2) .
    cq P1 right-to P2 = false if x(P1) < x(P2) .
    eq midpoint(P1,P2) = point((x(P1) + x(P2)) / 2,(y(P1) + y(P2)) / 2) .
endo
```

B.3 A Specification of Line

```
obj LINE is
    protecting ANGLE .
    protecting POINT .
    sorts Point? Line Line-V .   *** Line-V for vertical line
    subsort Point < Point? .
    subsort Line-V < Line .
    op line : Point Point -> Line .
    op vline : Point Point -> Line-V .
    op s-point1-Line : Line -> Point .
    op s-point2-Line : Line -> Point .
    op s-point1-VLine : Line-V -> Point .
    op s-point2-VLine : Line-V -> Point .
    op point-on-line? : Point Line -> Bool .
    op point-on-vline? : Point Line-V -> Bool .
    op parallel : -> Point? .
    op intersect : Line Line -> Point [memo] .
    vars P P1 P2 : Point .
    vars L L1 L2 : Line .
    var LV : Line-V .
    cq line(P1,P2) = vline(P1,P2)  if  x(P1) == x(P2) .
    eq s-point1-Line(line(P1, P2)) = P1 .
    eq s-point2-Line(line(P1, P2)) = P2 .
    cq s-point1-VLine(vline(P1,P2)) = P1  if  x(P1) == x(P2) .
    cq s-point2-VLine(vline(P1,P2)) = P2  if  x(P1) == x(P2) .
*** L1 is not parallel to L2 and L1, L2 are both general
    cq intersect(L1,L2) = point
                (((x(s-point2-Line(L1)) * y(s-point1-Line(L1)) -
                   x(s-point1-Line(L1)) * y(s-point2-Line(L1))) *
```

```
                (x(s-point2-Line(L2)) - x(s-point1-Line(L2))) -
                (x(s-point2-Line(L2)) * y(s-point1-Line(L2)) -
                 x(s-point1-Line(L2)) * y(s-point2-Line(L2))) *
                (x(s-point2-Line(L1)) - x(s-point1-Line(L1)))) /
               ((x(s-point2-Line(L1)) - x(s-point1-Line(L1))) *
                (y(s-point2-Line(L2)) - y(s-point1-Line(L2))) -
                (y(s-point2-Line(L1)) - y(s-point1-Line(L1))) *
                (x(s-point2-Line(L2)) - x(s-point1-Line(L2)))),
              ((y(s-point2-Line(L2)) - y(s-point1-Line(L2))) /
               (x(s-point2-Line(L2)) - x(s-point1-Line(L2)))) *
             (((x(s-point2-Line(L1)) * y(s-point1-Line(L1)) -
                x(s-point1-Line(L1)) * y(s-point2-Line(L1))) *
               (x(s-point2-Line(L2)) - x(s-point1-Line(L2))) -
               (x(s-point2-Line(L2)) * y(s-point1-Line(L2)) -
                x(s-point1-Line(L2)) * y(s-point2-Line(L2))) *
               (x(s-point2-Line(L1)) - x(s-point1-Line(L1)))) /
              ((x(s-point2-Line(L1)) - x(s-point1-Line(L1))) *
               (y(s-point2-Line(L2)) - y(s-point1-Line(L2))) -
               (y(s-point2-Line(L1)) - y(s-point1-Line(L1))) *
               (x(s-point2-Line(L2)) - x(s-point1-Line(L2))))) +
               (x(s-point2-Line(L2)) * y(s-point1-Line(L2)) -
                x(s-point1-Line(L2)) * y(s-point2-Line(L2))) /
               (x(s-point2-Line(L2)) - x(s-point1-Line(L2))))    if
            x(s-point2-Line(L1)) =/= x(s-point1-Line(L1)) and
            x(s-point2-Line(L2)) =/= x(s-point1-Line(L2)) and
            x(s-point2-VLine(L1)) =/= x(s-point1-VLine(L1)) and
            x(s-point2-VLine(L2)) =/= x(s-point1-VLine(L2)) and
      angle(x(s-point1-Line(L1)),y(s-point1-Line(L1)),
            x(s-point2-Line(L1)),y(s-point2-Line(L1))) -
      angle(x(s-point1-Line(L2)),y(s-point1-Line(L2)),
            x(s-point2-Line(L2)),y(s-point2-Line(L2))) =/= 0.0 and
      angle(x(s-point1-Line(L1)),y(s-point1-Line(L1)),
            x(s-point2-Line(L1)),y(s-point2-Line(L1))) -
      angle(x(s-point1-Line(L2)),y(s-point1-Line(L2)),
            x(s-point2-Line(L2)),y(s-point2-Line(L2))) =/= 180 and
      angle(x(s-point1-Line(L1)),y(s-point1-Line(L1)),
            x(s-point2-Line(L1)),y(s-point2-Line(L1))) -
      angle(x(s-point1-Line(L2)),y(s-point1-Line(L2)),
            x(s-point2-Line(L2)),y(s-point2-Line(L2))) =/= -180 .
*** L1 is vertical, L2 is general
    cq intersect(L1,L2) = point(x(s-point1-VLine(L1)),
             (y(s-point1-Line(L2)) * (x(s-point2-Line(L2)) -
              x(s-point1-VLine(L1))) + y(s-point2-Line(L2)) *
             (x(s-point1-VLine(L1)) - x(s-point1-Line(L2)))) /
             (x(s-point2-Line(L2)) - x(s-point1-Line(L2)))) if
              x(s-point2-VLine(L1)) == x(s-point1-VLine(L1)) and
              x(s-point2-Line(L2)) =/= x(s-point1-Line(L2)) .
*** L1 is general, L2 is vertical
    cq intersect(L1,L2) = point(x(s-point1-VLine(L2)),
                 (y(s-point1-Line(L1)) * (x(s-point2-Line(L1)) -
```

```
                    x(s-point1-VLine(L2))) + y(s-point2-Line(L1)) *
                   (x(s-point1-VLine(L2)) - x(s-point1-Line(L1)))) /
                   (x(s-point2-Line(L1)) - x(s-point1-Line(L1)))) if
                    x(s-point2-Line(L1)) =/= x(s-point1-Line(L1)) and
                    x(s-point2-VLine(L2)) == x(s-point1-VLine(L2)) .
*** Intersection of two parallel lines
    cq intersect(L1,L2) = parallel    if
            angle(x(s-point1-Line(L1)),y(s-point1-Line(L1)),
                  x(s-point2-Line(L1)),y(s-point2-Line(L1))) -
            angle(x(s-point1-Line(L2)),y(s-point1-Line(L2)),
                  x(s-point2-Line(L2)),y(s-point2-Line(L2))) == 0.0 or
            angle(x(s-point1-Line(L1)),y(s-point1-Line(L1)),
                  x(s-point2-Line(L1)),y(s-point2-Line(L1))) -
            angle(x(s-point1-Line(L2)),y(s-point1-Line(L2)),
                  x(s-point2-Line(L2)),y(s-point2-Line(L2))) == 180.0 or
           angle(x(s-point1-Line(L1)),y(s-point1-Line(L1)),
                 x(s-point2-Line(L1)),y(s-point2-Line(L1))) -
           angle(x(s-point1-Line(L2)),y(s-point1-Line(L2)),
                 x(s-point2-Line(L2)),y(s-point2-Line(L2))) == -180.0 .
*** If a point is on a line? L is general
    cq point-on-line?(P,L) = true  if
                             (y(P) - y(s-point1-Line(L))) /
                             (x(P) - x(s-point1-Line(L))) ==
                             (y(s-point2-Line(L)) - y(P)) /
                             (x(s-point2-Line(L)) - x(P)) .
    cq point-on-line?(P,L) = false  if
                             (y(P) - y(s-point1-Line(L))) /
                             (x(P) - x(s-point1-Line(L))) =/=
                             (y(s-point2-Line(L)) - y(P)) /
                             (x(s-point2-Line(L)) - x(P)) .
*** If a point is on a vertical line? L is vertical
    cq point-on-vline?(P,LV) = true  if
                              x(P) == x(s-point1-VLine(LV)) and
                              x(P) == x(s-point2-VLine(LV)) .
    cq point-on-vline?(P,LV) = false if
                              x(P) =/= x(s-point1-VLine(LV)) or
                              x(P) =/= x(s-point2-VLine(LV)) .
endo
```

Appendix C

Joint Abstraction of $\mathcal{SIS}$: An OBJ3 Simulation

C.1 A specification of the Enclosure method

```
obj PIVOT is
  protecting POINT *
  (sort Point to Pivot, op point to pivot) .
  op s-f1-Pivot : Pivot -> Float .    *** select 1st float in Pivot
  op s-f2-Pivot : Pivot -> Float .    *** select 2nd float in Pivot
  vars F1 F2 : Float .
  eq s-f1-Pivot(pivot(F1,F2)) = F1 .
  eq s-f2-Pivot(pivot(F1,F2)) = F2 .
endo

obj STEM is
  protecting PIVOT .
  protecting LINE-PA .
  sort Stem .
  subsort Stem < Line-pa .
  op stem : Pivot Degree -> Stem .
  op s-pivot-Stem : Stem -> Pivot .
  op s-angle-Stem : Stem -> Degree .
  op to-line : Stem -> Line .
  var P : Pivot .
  var A : Degree .
  vars F1 F2 : Float .
  eq s-pivot-Stem(stem(P,A)) = P .
  eq s-angle-Stem(stem(P,A)) = A .
  eq to-line(stem(pivot(F1,F2),A)) = to-line(line-pa(point(F1,F2),A)) .
endo

obj NODE is
  protecting STEM .
  sort Node .
  op node : Stem Float -> Node .
```

```
  op s-pivot-Node : Node -> Pivot .
  op s-stem-Node : Node -> Stem .
  op s-dist-Node : Node -> Float .
  op node-to-point : Node -> Point [memo] .
  var P : Pivot .
  var A : Degree .
  var S : Stem .
  var D : Float .
  var N : Node .
  eq s-pivot-Node(node(stem(P,A),D)) = P .
  eq s-stem-Node(node(S,D)) = S .
  eq s-dist-Node(node(S,D)) = D .
  var F1 F2 D : Float .
  eq node-to-point(N) = point
        (s-dist-Node(N) * cos(d-to-r(s-angle-Stem(s-stem-Node(N)))) +
         s-f1-Pivot(s-pivot-Stem(s-stem-Node(N))),
         s-dist-Node(N) * sin(d-to-r(s-angle-Stem(s-stem-Node(N)))) +
         s-f2-Pivot(s-pivot-Stem(s-stem-Node(N)))) .
endo

obj EDGE is
  protecting POINT-LL .
  protecting SEGMENT .
  protecting NODE .
  sort Edge .
  subsort Edge < Segment .
  op edge : Node Node -> Edge .
  op s-pivot-Edge : Edge -> Pivot .
  op s-node1-Edge : Edge -> Node .
  op s-node2-Edge : Edge -> Node .
  op c-node : Edge Stem -> Node .
  op length-of-edge : Edge -> Float .
  op area-by-edge : Edge -> Float .
  var P : Pivot .
  vars N1 N2 : Node .
  vars F1 F2 D1 D2 : Float .
  var A A1 A2 : Degree .
  eq s-pivot-Edge(edge(node(stem(P,A1),D1),node(stem(P,A2),D2))) = P .
  eq s-node1-Edge(edge(N1,N2)) = N1 .
  eq s-node2-Edge(edge(N1,N2)) = N2 .
  eq c-node(edge(N1,N2),stem(pivot(F1,F2),A)) =
     node(stem(pivot(F1,F2),A),distance-pp(mkpoint-ll(line(node-to-point(N1),
     node-to-point(N2)),to-line(stem(pivot(F1,F2),A))),point(F1,F2))) .
  eq length-of-edge(edge(N1,N2)) = length(segment(node-to-point(N1),
                                   node-to-point(N2))) .
  eq area-by-edge(edge(node(stem(pivot(F1,F2),A1),D1),
                       node(stem(pivot(F1,F2),A2),D2))) =
                       abs(0.5 * D1 * D2 * sin(d-to-r(A2 - A1))) .
endo
```

```
obj ENCLO is
  protecting INT .
  protecting EDGE .
  sorts Enclo NeEnclo .
  subsorts Edge < NeEnclo < Enclo .
  op nil : -> Enclo .
  op __ : Enclo Enclo -> Enclo [assoc id: nil prec 9] .
  op __ : Edge Enclo -> NeEnclo [assoc prec 9] .
  op head_ : NeEnclo -> Edge .
  op tail_ : NeEnclo -> Enclo .
  op visible?_ : Enclo -> Bool .
  op no-of-side_ : NeEnclo -> Int .
  op eval-length_ : NeEnclo -> Float .
  op eval-area_ : NeEnclo -> Float .
  var E : Edge .
  var N : NeEnclo .
  var A1 A2 : Degree .
  var D1 D2 : Float .
  eq head(E N) = E .
  eq tail(E N) = N .
  eq visible? N = N == nil .
  eq no-of-side E = 1 .
  eq no-of-side(E N) = 1 + no-of-side N .
  eq eval-length E = length-of-edge(E) .
  eq eval-length(E N) = length-of-edge(E) + eval-length N .
  eq eval-area E = area-by-edge(E) .
  eq eval-area(E N) = area-by-edge(E) + eval-area N .
endo
```

C.2 A specification of the Opening method

```
obj VP is
    protecting POINT *
    (sort Point to Vp, op point to vp, op x to s-f1-Vp,
     op y to s-f2-Vp) .
endo

obj VT is
    protecting VP *
    (sort Vp to Vt, op vp to vt, op s-f1-Vp to s-f1-Vt,
     op s-f2-Vp to s-f2-Vt)  .
endo

obj VL is
    protecting VP .
    protecting VT .
    protecting LINE .
    sort Vl .
    op vl : Vp Vt -> Vl .
    op s-vp-Vl : Vl -> Vp .
```

```
    op s-vt-Vl : Vl -> Vt .
    op point-on-vl? : Point Vl -> Bool .
    var P : Point .
    var P1 : Vp .
    var P2 : Vt .
    var V : Vl .
    eq s-vp-Vl(vl(P1,P2)) = P1 .
    eq s-vt-Vl(vl(P1,P2)) = P2 .
    cq point-on-vl?(P,V) = true  if  point-on-line?
            (P,line(point(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V))),
                    point(s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))))) .
    cq point-on-vl?(P,V) = false if  not  point-on-line?
            (P,line(point(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V))),
                    point(s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))))) .
endo

obj L-VL is
  protecting VP .
  protecting LINE-PA .
  sort L-vl .
  op l-vl : Vp Degree -> L-vl .
  op s-vp-Lvl : L-vl -> Vp .
  op s-angle-Lvl : L-vl -> Degree .
  op to-line : L-vl -> Line .
  vars F1 F2 : Float .
  var P : Vp .
  var A : Degree .
  var L : L-vl .
  eq s-vp-Lvl(l-vl(P,A)) = P .
  eq s-angle-Lvl(l-vl(P,A)) = A .
  eq to-line(l-vl(vp(F1,F2),A)) = to-line(line-pa(point(F1,F2),A)) .
endo

obj R-VL is
  protecting L-VL *
  (sort L-vl to R-vl, op l-vl to r-vl, op s-vp-Lvl to s-vp-Rvl,
   op s-angle-Lvl to s-angle-Rvl) .
endo

obj SL is
  sort Sl .
  protecting VL .
  protecting LINE-PA .
  op sl : Vl Float -> Sl .
  op point-sl : Vl Float -> Point .
  op angle-sl : Vl Float -> Degree .
  op sl-to-line : Sl -> Line .
  var V : Vl .
  var D : Float .
  cq point-sl(V,D) = point(s-f1-Vp(s-vp-Vl(V)) +
```

```
        D * cos(d-to-r(angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                             s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V)))),
                             s-f2-Vp(s-vp-Vl(V)) +
        D * sin(d-to-r(angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                             s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))))))
        if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) < 90 .
cq point-sl(V,D) = point(s-f1-Vp(s-vp-Vl(V)), D + s-f2-Vp(s-vp-Vl(V)))
        if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) == 90.0 .
cq point-sl(V,D) = point(s-f1-Vp(s-vp-Vl(V)) -
        D * cos(d-to-r(180 - angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
        s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V)))))),s-f2-Vp(s-vp-Vl(V)) +
        D * sin(d-to-r(180 - angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
        s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))))))
        if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) > 90  and
            angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) < 180 .
cq point-sl(V,D) = point(s-f1-Vp(s-vp-Vl(V)) - D, s-f2-Vp(s-vp-Vl(V)))
        if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                 s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) == 180.0 .
cq point-sl(V,D) = point(s-f1-Vp(s-vp-Vl(V)) +
        D * cos(d-to-r(angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
            s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) - 180)),
            s-f2-Vp(s-vp-Vl(V)) +
        D * sin(d-to-r(angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
            s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) - 180)))
        if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) > 180   and
            angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) < 270 .
cq point-sl(V,D) = point(s-f1-Vp(s-vp-Vl(V)), s-f2-Vp(s-vp-Vl(V)) - D)
        if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) == 270.0 .
cq point-sl(V,D) = point(s-f1-Vp(s-vp-Vl(V)) +
        D * cos(d-to-r(360 - angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
            s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) - 180)),
            s-f2-Vp(s-vp-Vl(V)) +
        D * sin(d-to-r(360 - angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
            s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) - 180)))
        if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) > 270   and
            angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) < 360 .
cq point-sl(V,D) = point(s-f1-Vp(s-vp-Vl(V)) + D, s-f2-Vp(s-vp-Vl(V)))
        if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) == 0.0   or
            angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                  s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) == 360.0 .
```

```
  cq angle-sl(V,D) = 90 + angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                                s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V)))
          if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                    s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) < 270 .
  cq angle-sl(V,D) = angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                           s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) - 270
          if  angle(s-f1-Vp(s-vp-Vl(V)),s-f2-Vp(s-vp-Vl(V)),
                    s-f1-Vt(s-vt-Vl(V)),s-f2-Vt(s-vt-Vl(V))) > 270 .
  eq sl-to-line(sl(V,D)) = to-line(line-pa(point-sl(V,D),angle-sl(V,D))) .
endo

obj ALPHA is
  protecting LINE .
  protecting L-VL .
  protecting SL .
  sort Alpha Alpha' .
  op isAlpha : Sl L-vl -> Alpha .      *** Alpha: constructive form
  op s-sl-Alpha : Alpha -> Sl .
  op s-lvl-Alpha : Alpha -> L-vl .
  op alpha : Sl L-vl -> Alpha' .       *** Alpha': constructed form
  op mkAlpha' : Float Float -> Alpha' .
  op alpha1 : Sl L-vl -> Float .
  op alpha2 : Sl L-vl -> Float .
  op s-alpha'1-Alpha' : Alpha' -> Float .
  op s-alpha'2-Alpha' : Alpha' -> Float .
  op translate-A : Alpha -> Alpha' [memo] .
  var L : L-vl .
  var S : Sl .
  var F1 F2 F3 F4 D : Float .
  var A : Degree .
  var A' : Alpha' .
  eq s-sl-Alpha(isAlpha(S,L)) = S .
  eq s-lvl-Alpha(isAlpha(S,L)) = L .
  eq alpha(S,L) = mkAlpha'(alpha1(S,L),alpha2(S,L)) .
  eq alpha1(S,L) = x(intersect(sl-to-line(S),to-line(L))) .
  eq alpha2(S,L) = y(intersect(sl-to-line(S),to-line(L))) .
  eq s-alpha'1-Alpha'(mkAlpha'(F1,F2)) = F1 .
  eq s-alpha'2-Alpha'(mkAlpha'(F1,F2)) = F2 .
  eq translate-A(isAlpha(S,L)) = alpha(S,L) .
endo

obj BETA is
  protecting LINE .
  protecting R-VL .
  protecting SL .
  sort Beta Beta' .
  op isBeta : Sl R-vl -> Beta .     *** Beta: constructive form
  op s-sl-Beta : Beta -> Sl .
  op s-rvl-Beta : Beta -> R-vl .
  op beta : Sl R-vl -> Beta' .      *** Beta': constructed form
```

```
  op mkBeta' : Float Float -> Beta' .
  op beta1 : Sl R-vl -> Float .
  op beta2 : Sl R-vl -> Float .
  op s-beta'1-Beta' : Beta' -> Float .
  op s-beta'2-Beta' : Beta' -> Float .
  op translate-B : Beta -> Beta' [memo] .
  var R : R-vl .
  var S : Sl .
  var F1 F2 F3 F4 D : Float .
  var A : Degree .
  var B : Beta' .
  eq s-sl-Beta(isBeta(S,R)) = S .
  eq s-rvl-Beta(isBeta(S,R)) = R .
  eq beta(S,R) = mkBeta'(beta1(S,R),beta2(S,R)) .
  eq beta1(S,R) = x(intersect(sl-to-line(S),to-line(R))) .
  eq beta2(S,R) = y(intersect(sl-to-line(S),to-line(R))) .
  eq s-beta'1-Beta'(mkBeta'(F1,F2)) = F1 .
  eq s-beta'2-Beta'(mkBeta'(F1,F2)) = F2 .
  eq translate-B(isBeta(S,R)) = beta(S,R) .
endo

obj FENE is
  protecting ALPHA .
  protecting BETA .
  sorts Fene Fene' .
  op isFene : Alpha Beta -> Fene .      *** Fene: constructive form
  op fene : Alpha Beta -> Fene' .
  op mkFene' : Alpha' Beta' -> Fene' . *** Fene': constructed form
  op s-alpha-Fene : Fene -> Alpha .
  op s-beta-Fene : Fene -> Beta .
  ops s-alpha'1-Fene s-alpha'2-Fene : Fene -> Float .
  ops s-beta'1-Fene s-beta'2-Fene  : Fene -> Float .
  ops s-alpha'1-Fene' s-alpha'2-Fene' : Fene' -> Float .
  ops s-beta'1-Fene' s-beta'2-Fene' : Fene' -> Float .
  op s-vp-Fene : Fene -> Vp .
  op translate-F : Fene -> Fene' [memo] .
  op width-of-fene : Fene -> Float .
  op width-of-fene' : Fene' -> Float .
  var L : L-vl .        var R : R-vl .
  var S : Sl .            var A : Alpha .
  var B : Beta .        var F : Fene .
  var F' : Fene' .      vars F1 F2 F3 F4 : Float .
  eq s-alpha-Fene(isFene(A,B)) = A .
  eq s-beta-Fene(isFene(A,B)) = B .
  eq s-vp-Fene(F) = s-vp-Lvl(s-lvl-Alpha(s-alpha-Fene(F))) .
  eq fene(A,B) = mkFene'(translate-A(A),translate-B(B)) .
  eq translate-F(isFene(A,B)) = mkFene'(translate-A(A),translate-B(B)) .
  eq s-alpha'1-Fene(F) = alpha1(s-sl-Alpha(s-alpha-Fene(F)),
                                s-lvl-Alpha(s-alpha-Fene(F))) .
  eq s-alpha'2-Fene(F) = alpha2(s-sl-Alpha(s-alpha-Fene(F)),
```

```
                                s-lvl-Alpha(s-alpha-Fene(F))) .
  eq s-beta'1-Fene(F) = beta1(s-sl-Beta(s-beta-Fene(F)),
                              s-rvl-Beta(s-beta-Fene(F))) .
  eq s-beta'2-Fene(F) = beta2(s-sl-Beta(s-beta-Fene(F)),
                              s-rvl-Beta(s-beta-Fene(F))) .
  eq s-alpha'1-Fene'(mkFene'(mkAlpha'(F1,F2),mkBeta'(F3,F4))) = F1 .
  eq s-alpha'2-Fene'(mkFene'(mkAlpha'(F1,F2),mkBeta'(F3,F4))) = F2 .
  eq s-beta'1-Fene'(mkFene'(mkAlpha'(F1,F2),mkBeta'(F3,F4)))  = F3 .
  eq s-beta'2-Fene'(mkFene'(mkAlpha'(F1,F2),mkBeta'(F3,F4)))  = F4 .
  eq width-of-fene'(F') = distance-pp(point(s-alpha'1-Fene'(F'),
                                            s-alpha'2-Fene'(F')),
                                      point(s-beta'1-Fene'(F'),
                                            s-beta'2-Fene'(F'))) .
  eq width-of-fene(F) = width-of-fene'(translate-F(F)) .
endo
```

C.3 A specification of the spatial operation On-Cutting

```
obj ON-CUT is
  sort Portion .
  protecting EDGE .
  protecting FENE .
  op mkPortion : Edge Fene -> Portion .
  op on-cut : Edge Fene' Edge -> Portion .
  op reconstruct-Node : Alpha Pivot -> Node [memo] .
  op reconstruct-Node : Beta Pivot -> Node [memo] .
  var E : Edge .  var F : Fene .  var A : Alpha .
  var B : Beta .  var P : Pivot .
  cq mkPortion(E,F) = on-cut
     (edge(s-node1-Edge(E),reconstruct-Node(s-alpha-Fene(F),s-pivot-Edge(E))),
      translate-F(F),
      edge(reconstruct-Node(s-beta-Fene(F),s-pivot-Edge(E)),s-node2-Edge(E)))
     if  point-on-segment?
     (point(s-alpha'1-Fene(F),s-alpha'2-Fene(F)),
      segment(node-to-point(s-node1-Edge(E)),
              node-to-point(s-node2-Edge(E))))
     and point-on-segment?
     (point(s-beta'1-Fene(F),s-beta'2-Fene(F)),
      segment(node-to-point(s-node1-Edge(E)),
              node-to-point(s-node2-Edge(E)))) .
  eq reconstruct-Node(A,P) = node(stem(P,angle(s-f1-Pivot(P),s-f2-Pivot(P),
                                               s-alpha'1-Alpha'(translate-A(A)),
                                               s-alpha'2-Alpha'(translate-A(A)))),
                           distance-pp(point(s-alpha'1-Alpha'(translate-A(A)),
                                               s-alpha'2-Alpha'(translate-A(A))),
                                         point(s-f1-Pivot(P),s-f2-Pivot(P)))) .
  eq reconstruct-Node(B,P) = node(stem(P,angle(s-f1-Pivot(P),s-f2-Pivot(P),
                                               s-beta'1-Beta'(translate-B(B)),
                                               s-beta'2-Beta'(translate-B(B)))),
                           distance-pp(point(s-beta'1-Beta'(translate-B(B)),
```

```
                                        s-beta'2-Beta'(translate-B(B))),
                                    point(s-f1-Pivot(P),s-f2-Pivot(P)))) .
endo
```

C.4 A specification of the spatial operation Out-Cutting

```
obj OUT-CUT is
  sorts Outcut1 Outcut2 .
  protecting LINE-PA .
  subsort Point < Outcut1 .
  subsort Point < Outcut2 .
  protecting ON-CUT .
  op mkoutcut1 : Fene Edge -> Outcut1 [memo] .
  op mkoutcut2 : Beta Edge -> Outcut2 [memo] .
  op reconstruct-Node : Outcut1 Pivot -> Node [memo] .
  op reconstruct-Node : Outcut2 Pivot -> Node [memo] .
  op out-cut : Edge Edge Fene' Edge Edge -> Portion .
  var E : Edge .    var F : Fene .    var A : Alpha .
  var B : Beta .    var P : Pivot .    var O1 : Outcut1 .
  var O2 : Outcut2 .
  eq mkoutcut1(F,E) = intersect(
       line(node-to-point(s-node1-Edge(E)),node-to-point(s-node2-Edge(E))),
       to-line(line-pa(point(s-alpha'1-Fene(F),s-alpha'2-Fene(F)),
                            90 + angle(s-alpha'1-Fene(F),s-alpha'2-Fene(F),
                                       s-beta'1-Fene(F),s-beta'2-Fene(F))))) .
 eq mkoutcut2(B,E) = perpend-P(
      line(node-to-point(s-node1-Edge(E)),node-to-point(s-node2-Edge(E))),
      point(s-beta'1-Beta'(translate-B(B)),s-beta'2-Beta'(translate-B(B)))) .

 eq reconstruct-Node(O1,P) = node(stem(P,angle(s-f1-Pivot(P),s-f2-Pivot(P),
                              x(O1),y(O1))),distance-pp(point(s-f1-Pivot(P),
                              s-f2-Pivot(P)),point(x(O1),y(O1)))) .

 eq reconstruct-Node(O2,P) = node(stem(P,angle(s-f1-Pivot(P),s-f2-Pivot(P),
                              x(O2),y(O2))),distance-pp(point(s-f1-Pivot(P),
                              s-f2-Pivot(P)),point(x(O2),y(O2)))) .
 cq mkPortion(E,F) = out-cut(
   edge(s-node1-Edge(E),
        reconstruct-Node(mkoutcut1(F,E),s-pivot-Edge(E))),
   edge(reconstruct-Node(mkoutcut1(F,E),s-pivot-Edge(E)),
        reconstruct-Node(s-alpha-Fene(F),s-pivot-Edge(E))),
   translate-F(F),
   edge(reconstruct-Node(s-beta-Fene(F),s-pivot-Edge(E)),
        reconstruct-Node(mkoutcut2(s-beta-Fene(F),E),s-pivot-Edge(E))),
   edge(reconstruct-Node(mkoutcut2(s-beta-Fene(F),E),s-pivot-Edge(E)),
        s-node2-Edge(E)))   if
    intersect(segment(point(s-alpha'1-Fene(F),s-alpha'2-Fene(F)),
                      point(s-f1-Vp(s-vp-Fene(F)),s-f2-Vp(s-vp-Fene(F)))),
              segment(node-to-point(s-node1-Edge(E)),
                      node-to-point(s-node2-Edge(E)))) =/= non-intersect  and
```

```
    intersect(segment(point(s-beta'1-Fene(F),s-beta'2-Fene(F)),
                      point(s-f1-Vp(s-vp-Fene(F)),s-f2-Vp(s-vp-Fene(F)))),
              segment(node-to-point(s-node1-Edge(E)),
                      node-to-point(s-node2-Edge(E)))) =/= non-intersect  and
    not point-on-segment?
    (point(s-alpha'1-Fene(F),s-alpha'2-Fene(F)),
    segment(node-to-point(s-node1-Edge(E)),
            node-to-point(s-node2-Edge(E))))   and
    not point-on-segment?
    (point(s-beta'1-Fene(F),s-beta'2-Fene(F)),
    segment(node-to-point(s-node1-Edge(E)),
            node-to-point(s-node2-Edge(E))))   and
    point-on-segment?
    (point(x(mkoutcut1(F,E)),
           y(mkoutcut1(F,E))),
    segment(node-to-point(s-node1-Edge(E)),node-to-point(s-node2-Edge(E)))) and
    point-on-segment?
    (point(x(mkoutcut2(s-beta-Fene(F),E)),
           y(mkoutcut2(s-beta-Fene(F),E))),
    segment(node-to-point(s-node1-Edge(E)),node-to-point(s-node2-Edge(E)))) .
endo
```

C.5 Two testing cases of runing On-Cutting

```
[Fene is dependent on Edge]
red mkPortion
    (edge(node(stem(pivot(20,7),150),20),node(stem(pivot(20,7),30),20)),
     isFene(isAlpha(sl(vl(vp(28,15),vt(28,70)),distance-pp(point(28,15),
            intersect(segment(point(28,15),point(28,70)),
                      segment(node-to-point(node(stem(pivot(20,7),150),20)),
                              node-to-point(node(stem(pivot(20,7),30),20))))))),
            l-vl(vp(28,15),150)),
            isBeta(sl(vl(vp(28,15),vt(28,70)),distance-pp(point(28,15),
            intersect(segment(point(28,15),point(28,70)),
                      segment(node-to-point(node(stem(pivot(20,7),150),20)),
                              node-to-point(node(stem(pivot(20,7),30),20))))))),
            r-vl(vp(28,15),30)))) .
rewrites: 3123
result Portion: on-cut(edge(node(stem(pivot(20.0,7.0),150.0),20.0),
    node(stem(pivot(20.0,7.0),65.6014388446695),10.98063632754477)),
    mkFene'(mkAlpha'(24.53589838486225,17.0),mkBeta'(31.46410161513775,
    17.0)),edge(node(stem(pivot(20.0,7.0),41.09780538932412),15.2126797718944),
    node(stem(pivot(20.0,7.0),30.0),20.0)))

[Edge is dependent on Fene]
red mkPortion
     (edge(node(stem(pivot(6,-2),120),distance-pp(point(6,-2),intersect(
      to-line(line-pa(point(6,-2),120)),
      line(point(s-alpha'1-Fene(isFene(isAlpha(sl(vl(vp(8,15),vt(10,24)),10),
             l-vl(vp(8,15),110)),isBeta(sl(vl(vp(8,15),vt(10,24)),10),
```

```
            r-vl(vp(8,15),65)))),s-alpha'2-Fene(isFene(isAlpha(sl(vl(vp(8,15),
                 vt(10,24)),10),l-vl(vp(8,15),120)),isBeta(sl(vl(vp(8,15),
                 vt(10,24)),10),r-vl(vp(8,15),65))))),
          point(s-beta'1-Fene(isFene(isAlpha(sl(vl(vp(8,15),vt(10,24)),10),
            l-vl(vp(8,15),110)),isBeta(sl(vl(vp(8,15),vt(10,24)),10),
            r-vl(vp(8,15),65)))),s-beta'2-Fene(isFene(isAlpha(sl(vl(vp(8,15),
                 vt(10,24)),10),l-vl(vp(8,15),110)),isBeta(sl(vl(vp(8,15),
                 vt(10,24)),10),r-vl(vp(8,15),65)))))))),
          node(stem(pivot(6,-2),60),distance-pp(point(6,-2),intersect(
      to-line(line-pa(point(6,-2),60)),
      line(point(s-alpha'1-Fene(isFene(isAlpha(sl(vl(vp(8,15),vt(10,24)),10),
            l-vl(vp(8,15),110)),isBeta(sl(vl(vp(8,15),vt(10,24)),10),
            r-vl(vp(8,15),65)))),s-alpha'2-Fene(isFene(isAlpha(sl(vl(vp(8,15),
                 vt(10,24)),10),l-vl(vp(8,15),110)),isBeta(sl(vl(vp(8,15),
                 vt(10,24)),10),r-vl(vp(8,15),65))))),
          point(s-beta'1-Fene(isFene(isAlpha(sl(vl(vp(8,15),
                 vt(10,24)),10),l-vl(vp(8,15),110)),isBeta(sl(vl(vp(8,15),
                 vt(10,24)),10),r-vl(vp(8,15),65)))),
                s-beta'2-Fene(isFene(isAlpha(sl(vl(vp(8,15),
                 vt(10,24)),10),l-vl(vp(8,15),110)),isBeta(sl(vl(vp(8,15),
                 vt(10,24)),10),r-vl(vp(8,15),65)))))))),
      isFene(isAlpha(sl(vl(vp(8,15),vt(10,24)),10),l-vl(vp(8,15),110)),
            isBeta(sl(vl(vp(8,15),vt(10,24)),10),r-vl(vp(8,15),65)))) .
rewrites: 1800
result Portion: on-cut(edge(node(stem(pivot(6.0,-2.0),120.0),39.1589631314743),
    node(stem(pivot(6.0,-2.0),94.17919679289533),28.22044196786174)),
    mkFene'(mkAlpha'(3.943404731090932,26.14540390119411),
            mkBeta'(12.3283103676397,24.2820915375166)),
    edge(node(stem(pivot(6.0,-2.0),76.46179418624334),27.03323598268579),
         node(stem(pivot(6.0,-2.0),60.0),28.33624811706596)))
```

C.6 A testing case of modelling a shape of *NeEnvelope*

```
reduce
mkPortion(edge(node(stem(pivot(8,8),180),6),node(stem(pivot(8,8),116.5),6.7)),
          isFene(isAlpha(sl(vl(vp(5,9.2),vt(2,12.5)),3.5),l-vl(vp(5,9.2),157)),
                 isBeta(sl(vl(vp(5,9.2),vt(2,12.5)),3.5),r-vl(vp(5,9.2),102))))
mkPortion(edge(node(stem(pivot(8,8),116.5),6.7),node(stem(pivot(8,8),63),6.7)),
          isFene(isAlpha(sl(vl(vp(8,12),vt(8,15)),2),l-vl(vp(8,12),125)),
                 isBeta(sl(vl(vp(8,12),vt(8,15)),2),r-vl(vp(8,12),57))))
mkPortion(edge(node(stem(pivot(8,8),63),6.7),node(stem(pivot(8,8),16),7.3)),
 isFene(isAlpha(sl(vl(vp(11.3,9.6),vt(13,12.3)),2.1),l-vl(vp(11.3,9.6),85)),
       isBeta(sl(vl(vp(11.3,9.6),vt(13,12.3)),2.1),r-vl(vp(11.3,9.6),25)))).

rewrites: 43974
result NeEnvelope: out-cut(edge(node(stem(pivot(8.0,8.0),180.0),6.0),
    node(stem(pivot(8.0,8.0),165.7054781942579),5.489453555997502)),
    edge(node(stem(pivot(8.0,8.0),165.7054781942579),5.489453555997502),
         node(stem(pivot(8.0,8.0),157.5466603358499),7.083998071357219)),
    mkFene'(mkAlpha'(1.453033627441351,10.70559789909348),
```

```
        mkBeta'(4.157401522521126,13.16411416734782)),
edge(node(stem(pivot(8.0,8.0),126.652800595202),6.436896627453744),
     node(stem(pivot(8.0,8.0),125.0090148647666),6.091765551238675)),
edge(node(stem(pivot(8.0,8.0),125.0090148647666),6.091765551238675),
     node(stem(pivot(8.0,8.0),116.5),6.7)))

on-cut(edge(node(stem(pivot(8.0,8.0),116.5),6.7),node(stem(pivot(8.0,8.0),
103.1377812784058),6.161263051214646)),
mkFene'(mkAlpha'(6.599584923580581,14.0),mkBeta'(9.298815186395021,14.0)),
edge(node(stem(pivot(8.0,8.0),77.78568458230568),6.138967412229058),

in-cut(edge(node(stem(pivot(8.0,8.0),63.0),6.7),
           node(stem(pivot(8.0,8.0),55.99997570300782),6.493251733335771)),
edge(node(stem(pivot(8.0,8.0),55.99997570300782),6.493251733335771),
     node(stem(pivot(8.0,8.0),48.42429683096491),5.282892834817734)),
mkFene'(mkAlpha'(11.5057754527503,11.95202418756034),
        mkBeta'(13.56435083222577,10.65588413381652)),
edge(node(stem(pivot(8.0,8.0),25.51530649798233),6.165688989589882),
     node(stem(pivot(8.0,8.0),25.4692059744215),6.771465950830739)),
edge(node(stem(pivot(8.0,8.0),25.4692059744215),6.771465950830739),
     node(stem(pivot(8.0,8.0),16.0),7.3))))
```

Appendix D

Joint Substantiation of $\mathcal{CGS}$: An OBJ3 Simulation

D.1 An Example of Parameterised Programming

D.1.1 A theory of alignment

```
th ALIGN is protecting LINE .
  sort Point .    *** the element
  sorts Comp1 Comp2 Comp3 Comp4 . *** four components
  sort Object .  *** an object of 4-component
  sort RefLine .    *** reference line
  subsort RefLine < Line .
  op comp1 : Point Point -> Comp1 .
  op comp2 : Point Point -> Comp2 .
  op comp3 : Point Point -> Comp3 .
  op comp4 : Point Point -> Comp4 .
  op select-P1-Comp1 : Comp1 -> Point .
  op select-P2-Comp1 : Comp1 -> Point .
  op select-P1-Comp2 : Comp2 -> Point .
  op select-P2-Comp2 : Comp2 -> Point .
  op select-P1-Comp3 : Comp3 -> Point .
  op select-P2-Comp3 : Comp3 -> Point .
  op select-P1-Comp4 : Comp4 -> Point .
  op select-P2-Comp4 : Comp4 -> Point .
  op object : Comp1 Comp2 Comp3 Comp4 -> Object .
  op find-refline : Object -> RefLine [memo] .
  op is-refline : Point Point -> RefLine .
  op select-Comp1 : Object -> Comp1 .
  op select-Comp2 : Object -> Comp2 .
  op select-Comp3 : Object -> Comp3 .
  op select-Comp4 : Object -> Comp4 .
  op select-P1-RefLine : RefLine -> Point .
  op select-P2-RefLine : RefLine -> Point .
  vars P1 P2 : Point .
  var O : Object .
```

```
  var C1 : Comp1 . var C2 : Comp2 .
  var C3 : Comp3 . var C4 : Comp4 .
  eq select-P1-Comp1(comp1(P1,P2)) = P1 .
  eq select-P2-Comp1(comp1(P1,P2)) = P2 .
  eq select-P1-Comp2(comp2(P1,P2)) = P1 .
  eq select-P2-Comp2(comp2(P1,P2)) = P2 .
  eq select-P1-Comp3(comp3(P1,P2)) = P1 .
  eq select-P2-Comp3(comp3(P1,P2)) = P2 .
  eq select-P1-Comp4(comp4(P1,P2)) = P1 .
  eq select-P2-Comp4(comp4(P1,P2)) = P2 .
  eq select-Comp1(object(C1,C2,C3,C4)) = C1 .
  eq select-Comp2(object(C1,C2,C3,C4)) = C2 .
  eq select-Comp3(object(C1,C2,C3,C4)) = C3 .
  eq select-Comp4(object(C1,C2,C3,C4)) = C4 .
  eq select-P1-RefLine(is-refline(P1,P2)) = P1 .
  eq select-P2-RefLine(is-refline(P1,P2)) = P2 .
endth
```

D.1.2 A parameterised anchoring function

```
obj ANCHOR[X :: ALIGN] is
  sort Unit .
  sorts Anchor-pt BaseLine .
  subsorts BaseLine < Line .
  op anchor-pt : Float Float -> Anchor-pt .
  op baseline : Point Point -> BaseLine .
  op anchor : Object Anchor-pt BaseLine -> Unit .
  op isUnit : Object BaseLine Anchor-pt -> Unit .
  op is-anchored-on : BaseLine -> BaseLine .
  op at-anchor-pt : Anchor-pt -> Anchor-pt .
  op translation : Object Anchor-pt -> Object .
  op x : Anchor-pt -> Float .
  op y : Anchor-pt -> Float .
  var O : Object . var A : Anchor-pt . var B : BaseLine .
  vars F1 F2 : Float .
  eq x(anchor-pt(F1,F2)) = F1 .
  eq y(anchor-pt(F1,F2)) = F2 .
  eq translation(O,A) = object(comp1(
        point(x(select-P1-Comp1(select-Comp1(O))) +
              (x(A) - x(select-P1-RefLine(find-refline(O)))),
               y(select-P1-Comp1(select-Comp1(O))) +
              (y(A) - y(select-P1-RefLine(find-refline(O))))),
        point(x(select-P2-Comp1(select-Comp1(O))) +
              (x(A) - x(select-P2-RefLine(find-refline(O)))),
               y(select-P2-Comp1(select-Comp1(O))) +
              (y(A) - y(select-P2-RefLine(find-refline(O))))),
                                comp2(
        point(x(select-P1-Comp2(select-Comp2(O))) +
              (x(A) - x(select-P1-RefLine(find-refline(O)))),
               y(select-P1-Comp2(select-Comp2(O))) +
              (y(A) - y(select-P1-RefLine(find-refline(O))))),
```

```
        point(x(select-P2-Comp2(select-Comp2(O))) +
              (x(A) - x(select-P2-RefLine(find-refline(O)))),
               y(select-P2-Comp2(select-Comp2(O))) +
              (y(A) - y(select-P2-RefLine(find-refline(O))))),
                                comp3(
        point(x(select-P1-Comp3(select-Comp3(O))) +
              (x(A) - x(select-P1-RefLine(find-refline(O)))),
               y(select-P1-Comp3(select-Comp3(O))) +
              (y(A) - y(select-P1-RefLine(find-refline(O))))),
         point(x(select-P2-Comp3(select-Comp3(O))) +
              (x(A) - x(select-P2-RefLine(find-refline(O)))),
               y(select-P2-Comp3(select-Comp3(O))) +
              (y(A) - y(select-P2-RefLine(find-refline(O))))),
                                comp4(
         point(x(select-P1-Comp1(select-Comp1(O))) +
              (x(A) - x(select-P1-RefLine(find-refline(O)))),
               y(select-P1-Comp1(select-Comp1(O))) +
              (y(A) - y(select-P1-RefLine(find-refline(O))))),
         point(x(select-P2-Comp1(select-Comp1(O))) +
              (x(A) - x(select-P2-RefLine(find-refline(O)))),
               y(select-P2-Comp1(select-Comp1(O))) +
              (y(A) - y(select-P2-RefLine(find-refline(O)))))) .
  eq anchor(O,A,B) = isUnit(translation(O,A),
                            is-anchored-on(B),
                            at-anchor-pt(A)) .
endo
```

D.1.3 An object of rectangle

```
obj RECTANGLE is protecting SEGMENT .
  sort Rectangle .
  sorts S-side1 S-side2 L-side1 L-side2 .
  sort SC-Line .  *** Central Line on short side
  sort LC-Line .  *** Central Line on long side
  sort Area .   *** Area of a rectangle
  subsorts S-side1 S-side2 L-side1 L-side2 < Segment .
  subsort SC-Line LC-Line < Line .
  subsort Area < Float .
  op rec : S-side1 S-side2 L-side1 L-side2 -> Rectangle .
  op s-side1 : Point Point -> S-side1 .
  op s-side2 : Point Point -> S-side2 .
  op l-side1 : Point Point -> L-side1 .
  op l-side2 : Point Point -> L-side2 .
  op s-P1-S-side1 : S-side1 -> Point .
  op s-P2-S-side1 : S-side1 -> Point .
  op s-P1-S-side2 : S-side2 -> Point .
  op s-P2-S-side2 : S-side2 -> Point .
  op s-P1-L-side1 : L-side1 -> Point .
  op s-P2-L-side1 : L-side1 -> Point .
  op s-P1-L-side2 : L-side2 -> Point .
  op s-P2-L-side2 : L-side2 -> Point .
```

```
  op select-S-side1 : Rectangle -> S-side1 .
  op select-S-side2 : Rectangle -> S-side2 .
  op select-L-side1 : Rectangle -> L-side1 .
  op select-L-side2 : Rectangle -> L-side2 .
  op sc-line : Rectangle -> SC-Line .
  op lc-line : Rectangle -> LC-Line .
  op is-SC-Line : Point Point -> SC-Line .
  op is-LC-Line : Point Point -> LC-Line .
  op select-P1-SC-Line : SC-Line -> Point .
  op select-P2-SC-Line : SC-Line -> Point .
  op select-P1-LC-Line : LC-Line -> Point .
  op select-P2-LC-Line : LC-Line -> Point .
  op area : Rectangle -> Area .
  var S1 : S-side1 . var S2 : S-side2 .
  var L1 : L-side1 . var L2 : L-side2 .
  var R : Rectangle .
  vars P1 P2 : Point .
  eq s-P1-S-side1(s-side1(P1,P2)) = P1 .
  eq s-P2-S-side1(s-side1(P1,P2)) = P2 .
  eq s-P1-S-side2(s-side2(P1,P2)) = P1 .
  eq s-P2-S-side2(s-side2(P1,P2)) = P2 .
  eq s-P1-L-side1(l-side1(P1,P2)) = P1 .
  eq s-P2-L-side1(l-side1(P1,P2)) = P2 .
  eq s-P1-L-side2(l-side2(P1,P2)) = P1 .
  eq s-P2-L-side2(l-side2(P1,P2)) = P2 .
  eq select-S-side1(rec(S1,S2,L1,L2)) = S1 .
  eq select-S-side2(rec(S1,S2,L1,L2)) = S2 .
  eq select-L-side1(rec(S1,S2,L1,L2)) = L1 .
  eq select-L-side2(rec(S1,S2,L1,L2)) = L2 .
  eq sc-line(R) = is-SC-Line(mid-seg(
                      segment(s-P1-S-side1(select-S-side1(R)),
                              s-P2-S-side1(select-S-side1(R)))),
                             mid-seg(
                      segment(s-P1-S-side2(select-S-side2(R)),
                              s-P2-S-side2(select-S-side2(R))))) .
  eq lc-line(R) = is-LC-Line(mid-seg(
                      segment(s-P1-L-side1(select-L-side1(R)),
                              s-P2-L-side1(select-L-side1(R)))),
                             mid-seg(
                      segment(s-P1-L-side2(select-L-side2(R)),
                              s-P2-L-side2(select-L-side2(R))))) .
  eq select-P1-SC-Line(is-SC-Line(P1,P2)) = P1 .
  eq select-P2-SC-Line(is-SC-Line(P1,P2)) = P2 .
  eq select-P1-LC-Line(is-LC-Line(P1,P2)) = P1 .
  eq select-P2-LC-Line(is-LC-Line(P1,P2)) = P2 .
  eq area(R) = length(segment(s-P1-S-side1(select-S-side1(R)),
                              s-P2-S-side1(select-S-side1(R)))) *
               length(segment(s-P1-L-side1(select-L-side1(R)),
                              s-P2-L-side1(select-L-side1(R)))) .
endo
```

D.2 Wall Junction Design: A General Case

D.2.1 A module for junction construction

```
obj JOINING is protecting SEGMENT .
   sorts Join-rec Fixed-rec Translated-rec .
   sorts Origin Join-pt S-opp-pt C-opp-pt .
   sorts Join-seg1 Join-seg2 Fixed-seg1 Fixed-seg2 .
   sorts S-j-seg S-f-seg F-seg J-seg C-j-seg C-f-seg .
   sorts Join-side Fixed-side .
   sorts M-j-seg M-f-seg .
   sorts S-Junc C-Junc .
   subsorts Origin Join-pt S-opp-pt C-opp-pt < Point .
   subsorts Join-seg1 Join-side Join-seg2 < Segment .
   subsorts Fixed-seg1 Fixed-side Fixed-seg2 < Segment .
   subsort M-j-seg M-f-seg < Segment .
   op origin : Float Float -> Origin .
   op join-seg1 : Origin Point -> Join-seg1 .
   op join-seg2 : Point Point -> Join-seg2 .
   op join-side : Origin Point -> Join-side .
   op join-rec : Join-seg1 Join-side Join-seg2 -> Join-rec .
   op s-P1-Join-seg1 : Join-seg1 -> Point .
   op s-P2-Join-seg1 : Join-seg1 -> Point .
   op s-P1-Join-seg2 : Join-seg2 -> Point .
   op s-P2-Join-seg2 : Join-seg2 -> Point .
   op s-P1-Join-side : Join-side -> Point .
   op s-P2-Join-side : Join-side -> Point .
   op select-join-seg1 : Join-rec -> Join-seg1 .
   op select-join-side : Join-rec -> Join-side .
   op select-join-seg2 : Join-rec -> Join-seg2 .
   op join-pt : Float Float -> Join-pt .
   op fixed-seg1 : Join-pt Point -> Fixed-seg1 .
   op fixed-seg2 : Point Point -> Fixed-seg2 .
   op fixed-side : Join-pt Point -> Fixed-side .
   op fixed-rec : Fixed-seg1 Fixed-side Fixed-seg2 -> Fixed-rec .
   op s-P1-Fixed-seg1 : Fixed-seg1 -> Point .
   op s-P2-Fixed-seg1 : Fixed-seg1 -> Point .
   op s-P1-Fixed-seg2 : Fixed-seg2 -> Point .
   op s-P2-Fixed-seg2 : Fixed-seg2 -> Point .
   op s-P1-Fixed-side : Fixed-side -> Point .
   op s-P2-Fixed-side : Fixed-side -> Point .
   op select-fixed-seg1 : Fixed-rec -> Fixed-seg1 .
   op select-fixed-seg2 : Fixed-rec -> Fixed-seg2 .
   op select-fixed-side : Fixed-rec -> Fixed-side .
   op translated-rec : Segment Segment Segment -> Translated-rec .
   op trans-seg1 : Join-rec Join-pt -> Segment [memo] .
   op trans-seg2 : Join-rec Join-pt -> Segment [memo] .
   op trans-side : Join-rec Join-pt -> Segment [memo] .
   op select-Trans-seg1 : Translated-rec -> Segment .
   op select-Trans-seg2 : Translated-rec -> Segment .
   op select-Trans-side : Translated-rec -> Segment .
```

```
op translation : Join-rec Join-pt -> Translated-rec [memo] .
op s-opp-pt : Float Float -> S-opp-pt .
op c-opp-pt : Float Float -> C-opp-pt .
op s-j-seg : S-opp-pt Point -> S-j-seg .
op s-f-seg : S-opp-pt Point -> S-f-seg .
op f-seg : Join-pt Point -> F-seg .
op j-seg : Join-pt Point -> J-seg .
op c-j-seg : C-opp-pt Point -> C-j-seg .
op c-f-seg : C-opp-pt Point -> C-f-seg .
op sm-j-seg : S-opp-pt Join-pt -> M-j-seg .
op sm-f-seg : Join-pt S-opp-pt -> M-f-seg .
op cm-j-seg : C-opp-pt Join-pt -> M-j-seg .
op cm-f-seg : Join-pt C-opp-pt -> M-f-seg .
op joining-by-s : Join-rec Fixed-rec -> S-Junc .
op joining-by-c : Join-rec Fixed-rec -> C-Junc .
op is-S-Junc : S-j-seg M-j-seg J-seg
               S-f-seg M-f-seg F-seg -> S-Junc .
op is-C-Junc : J-seg M-j-seg C-j-seg
               F-seg M-f-seg C-f-seg -> C-Junc .
op select-trans-seg1 : Translated-rec -> Segment .
op select-trans-seg2 : Translated-rec -> Segment .
op select-trans-side : Translated-rec -> Segment .
op s-P1-trans-seg1 : Segment -> Point .
op s-P2-trans-seg1 : Segment -> Point .
op s-P1-trans-side : Segment -> Point .
op s-P2-trans-side : Segment -> Point .
op s-P1-trans-seg2 : Segment -> Point .
op s-P2-trans-seg2 : Segment -> Point .
op s-Join-pt : Fixed-rec -> Join-pt [memo] .
op s1-S-Junc : S-Junc -> S-j-seg .
op s2-S-Junc : S-Junc -> M-j-seg .
op s3-S-Junc : S-Junc -> J-seg .
op s4-S-Junc : S-Junc -> S-f-seg .
op s5-S-Junc : S-Junc -> M-f-seg .
op s6-S-Junc : S-Junc -> F-seg .
op s1-C-Junc : C-Junc -> J-seg .
op s2-C-Junc : C-Junc -> M-j-seg .
op s3-C-Junc : C-Junc -> C-j-seg .
op s4-C-Junc : C-Junc -> F-seg .
op s5-C-Junc : C-Junc -> M-f-seg .
op s6-C-Junc : C-Junc -> C-f-seg .
vars F1 F2 : Float . var O : Origin . vars P P1 P2 : Point .
var JREC : Join-rec . var FREC : Fixed-rec .
var JPT  : Join-pt . vars SEG1 SEG2 SEG3 : Segment .
var FSEG1 : Fixed-seg1 . var FSIDE : Fixed-side .
var FSEG2 : Fixed-seg2 . var JSEG1 : Join-seg1 .
var JSIDE : Join-side . var JSEG2 : Join-seg2 .
var SJ : S-j-seg . var MJ : M-j-seg . var J : J-seg .
var SF : S-f-seg . var MF : M-f-seg . var F : F-seg .
var CJ : C-j-seg . var CF : C-f-seg .
```

```
eq x(origin(F1,F2)) = F1 . eq x(join-pt(F1,F2)) = F1 .
eq y(origin(F1,F2)) = F2 . eq y(join-pt(F1,F2)) = F2 .
eq s-P1-Join-seg1(join-seg1(O,P)) = O .
eq s-P2-Join-seg1(join-seg1(O,P)) = P .
eq s-P1-Join-seg2(join-seg2(P1,P2)) = P1 .
eq s-P2-Join-seg2(join-seg2(P1,P2)) = P2 .
eq s-P1-Join-side(join-side(O,P)) = O .
eq s-P2-Join-side(join-side(O,P)) = P .
eq select-join-seg1(join-rec(JSEG1,JSIDE,JSEG2)) = JSEG1 .
eq select-join-side(join-rec(JSEG1,JSIDE,JSEG2)) = JSIDE .
eq select-join-seg2(join-rec(JSEG1,JSIDE,JSEG2)) = JSEG2 .
eq select-fixed-seg1(fixed-rec(FSEG1,FSIDE,FSEG2)) = FSEG1 .
eq select-fixed-side(fixed-rec(FSEG1,FSIDE,FSEG2)) = FSIDE .
eq select-fixed-seg2(fixed-rec(FSEG1,FSIDE,FSEG2)) = FSEG2 .
eq s-P1-Fixed-seg1(fixed-seg1(JPT,P)) = JPT .
eq s-P2-Fixed-seg1(fixed-seg1(JPT,P)) = P .
eq s-P1-Fixed-side(fixed-side(JPT,P)) = JPT .
eq s-P2-Fixed-side(fixed-side(JPT,P)) = P .
eq s-P1-Fixed-seg2(fixed-seg2(P1,P2)) = P1 .
eq s-P2-Fixed-seg2(fixed-seg2(P1,P2)) = P2 .
eq select-Trans-seg1(translated-rec(SEG1,SEG2,SEG3)) = SEG1 .
eq select-Trans-seg2(translated-rec(SEG1,SEG2,SEG3)) = SEG3 .
eq select-Trans-side(translated-rec(SEG1,SEG2,SEG3)) = SEG2 .
eq s1-S-Junc(is-S-Junc(SJ,MJ,J,SF,MF,F)) = SJ .
eq s2-S-Junc(is-S-Junc(SJ,MJ,J,SF,MF,F)) = MJ .
eq s3-S-Junc(is-S-Junc(SJ,MJ,J,SF,MF,F)) = J .
eq s4-S-Junc(is-S-Junc(SJ,MJ,J,SF,MF,F)) = SF .
eq s5-S-Junc(is-S-Junc(SJ,MJ,J,SF,MF,F)) = MF .
eq s6-S-Junc(is-S-Junc(SJ,MJ,J,SF,MF,F)) = F .
eq s1-C-Junc(is-C-Junc(J,MJ,CJ,F,MF,CF)) = J .
eq s2-C-Junc(is-C-Junc(J,MJ,CJ,F,MF,CF)) = MJ .
eq s3-C-Junc(is-C-Junc(J,MJ,CJ,F,MF,CF)) = CJ .
eq s4-C-Junc(is-C-Junc(J,MJ,CJ,F,MF,CF)) = F .
eq s5-C-Junc(is-C-Junc(J,MJ,CJ,F,MF,CF)) = MF .
eq s6-C-Junc(is-C-Junc(J,MJ,CJ,F,MF,CF)) = CF .

eq trans-seg1(JREC,JPT) = segment(point(x(JPT),y(JPT)),
    point(x(s-P2-Join-seg1(select-join-seg1(JREC))) +
         (x(JPT) - x(s-P1-Join-seg1(select-join-seg1(JREC)))),
          y(s-P2-Join-seg1(select-join-seg1(JREC))) +
         (y(JPT) - y(s-P1-Join-seg1(select-join-seg1(JREC)))))) .
eq trans-seg2(JREC,JPT) = segment(
    point(x(s-P1-Join-seg2(select-join-seg2(JREC))) +
         (x(JPT) - x(s-P1-Join-seg1(select-join-seg1(JREC)))),
          y(s-P1-Join-seg2(select-join-seg2(JREC))) +
         (y(JPT) - y(s-P1-Join-seg1(select-join-seg1(JREC))))),
    point(x(s-P2-Join-seg2(select-join-seg2(JREC))) +
         (x(JPT) - x(s-P1-Join-seg1(select-join-seg1(JREC)))),
          y(s-P2-Join-seg2(select-join-seg2(JREC))) +
         (x(JPT) - x(s-P1-Join-seg1(select-join-seg1(JREC)))))) .
```

```
   eq trans-side(JREC,JPT) = segment(point(x(JPT),y(JPT)),
       point(x(s-P2-Join-seg2(select-join-seg2(JREC))) +
            (x(JPT) - x(s-P1-Join-seg1(select-join-seg1(JREC)))),
             y(s-P2-Join-seg2(select-join-seg2(JREC))) +
            (y(JPT) - y(s-P1-Join-seg1(select-join-seg1(JREC)))))) .
   eq s-P1-trans-seg1(segment(P1,P2)) = P1 .
   eq s-P2-trans-seg1(segment(P1,P2)) = P2 .
   eq s-P1-trans-seg2(segment(P1,P2)) = P1 .
   eq s-P2-trans-seg2(segment(P1,P2)) = P2 .
   eq s-P1-trans-side(segment(P1,P2)) = P1 .
   eq s-P2-trans-side(segment(P1,P2)) = P2 .
   eq translation(JREC,JPT) = translated-rec(
            trans-seg1(JREC,JPT),trans-side(JREC,JPT),
            trans-seg2(JREC,JPT)) .
   eq s-Join-pt(fixed-rec(FSEG1,FSIDE,FSEG2)) =
          s-P1-Fixed-seg1(FSEG1) .

*** To make a junction by stretching
  cq joining-by-s(JREC,FREC) = is-S-Junc(
    s-j-seg(s-opp-pt(x(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC)))),
                  y(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC))))),
            s-P2-trans-seg2(select-Trans-seg2(translation(JREC,
            s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),

    sm-j-seg(s-opp-pt(x(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC))))),
                     y(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
                  s-P2-trans-seg2(
```

```
   select-Trans-seg2(translation(JREC,
   s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
      segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
              s-P2-Fixed-seg2(select-fixed-seg2(FREC)))))),
   s-P1-Fixed-seg1(select-fixed-seg1(FREC))),

j-seg(s-P1-Fixed-seg1(select-fixed-seg1(FREC)),
      s-P2-trans-seg1(select-Trans-seg1(translation(JREC,
      s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),

s-f-seg(s-opp-pt(x(intersect(segment(s-P1-trans-seg2(
   select-Trans-seg2(translation(JREC,
   s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
              s-P2-trans-seg2(
   select-Trans-seg2(translation(JREC,
   s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
      segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
              s-P2-Fixed-seg2(select-fixed-seg2(FREC)))))),
              y(intersect(segment(s-P1-trans-seg2(
   select-Trans-seg2(translation(JREC,
   s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
              s-P2-trans-seg2(
   select-Trans-seg2(translation(JREC,
   s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
      segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
              s-P2-Fixed-seg2(select-fixed-seg2(FREC)))))),
        s-P2-Fixed-seg2(select-fixed-seg2(FREC))),

sm-f-seg(s-P1-Fixed-seg1(select-fixed-seg1(FREC)),
   s-opp-pt(x(intersect(segment(s-P1-trans-seg2(
   select-Trans-seg2(translation(JREC,
   s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
              s-P2-trans-seg2(
   select-Trans-seg2(translation(JREC,
   s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
      segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
              s-P2-Fixed-seg2(select-fixed-seg2(FREC)))))),
              y(intersect(segment(s-P1-trans-seg2(
   select-Trans-seg2(translation(JREC,
   s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
              s-P2-trans-seg2(
   select-Trans-seg2(translation(JREC,
   s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
      segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
              s-P2-Fixed-seg2(select-fixed-seg2(FREC)))))),

f-seg(s-P1-Fixed-seg1(select-fixed-seg1(FREC)),
      s-P2-Fixed-seg1(select-fixed-seg1(FREC))))       if
point-on-segment?(intersect(select-Trans-seg2(translation(JREC,
      s-P1-Fixed-seg1(select-fixed-seg1(FREC)))),
```

```
          select-fixed-seg2(FREC)),
          select-Trans-seg2(translation(JREC,
          s-P1-Fixed-seg1(select-fixed-seg1(FREC))))) =/= true and
    point-on-segment?(intersect(select-Trans-seg2(translation(JREC,
          s-P1-Fixed-seg1(select-fixed-seg1(FREC)))),
    select-fixed-seg2(FREC)),select-fixed-seg2(FREC)) =/= true .

*** To make a junction by contracting
 cq joining-by-c(JREC,FREC) = is-C-Junc(
    j-seg(s-P1-Fixed-seg1(select-fixed-seg1(FREC)),
          s-P2-trans-seg1(select-Trans-seg1(translation(JREC,
          s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),

    cm-j-seg(c-opp-pt(x(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC)))),
                  y(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC))))),
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))),

    c-j-seg(c-opp-pt(x(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC)))),
                  y(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC))))),
          s-P2-trans-seg2(select-Trans-seg2(translation(JREC,
          s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
```

```
    f-seg(s-P1-Fixed-seg1(select-fixed-seg1(FREC)),
          s-P2-Fixed-seg1(select-fixed-seg1(FREC))),

    cm-f-seg(s-P1-Fixed-seg1(select-fixed-seg1(FREC)),
       c-opp-pt(x(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC)))),
                  y(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC)))))),

    c-f-seg(c-opp-pt(x(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC)))),
                  y(intersect(segment(s-P1-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC))))),
                  s-P2-trans-seg2(
       select-Trans-seg2(translation(JREC,
       s-P1-Fixed-seg1(select-fixed-seg1(FREC)))))),
          segment(s-P1-Fixed-seg2(select-fixed-seg2(FREC)),
                  s-P2-Fixed-seg2(select-fixed-seg2(FREC))))),
          s-P2-Fixed-seg2(select-fixed-seg2(FREC))))     if
    point-on-segment?(intersect(select-Trans-seg2(translation(JREC,
          s-P1-Fixed-seg1(select-fixed-seg1(FREC)))),
    select-fixed-seg2(FREC)),select-Trans-seg2(translation(JREC,
          s-P1-Fixed-seg1(select-fixed-seg1(FREC))))) == true and
    point-on-segment?(intersect(select-Trans-seg2(translation(JREC,
          s-P1-Fixed-seg1(select-fixed-seg1(FREC)))),
          select-fixed-seg2(FREC)),
          select-Trans-seg2(translation(JREC,
          s-P1-Fixed-seg1(select-fixed-seg1(FREC))))) == true .
endo
```

D.2.2 A module for wall description

```
obj WALL is protecting JOINING .
  sorts Material Wall-Type .
  sorts F-face1 End1 End2 F-face2 J-face1 J-face2 .
  sorts Fixed-Wall Join-Wall Junc-Wall .
  op brick : -> Material .
  op block : -> Material .
  op concrete : -> Material .
  op internal-wall : -> Wall-Type .
  op external-wall : -> Wall-Type .
  op is-wall-type-of : Wall-Type -> Wall-Type .
  op is-material-of : Material -> Material .
  op is-SJ-face1 : S-j-seg -> J-face1 .
  op is-SF-face1 : S-f-seg -> F-face1 .
  op is-End1 : M-j-seg -> End1 .
  op is-End2 : M-f-seg -> End2 .
  op is-F-face2 : F-seg -> F-face2 .
  op is-J-face2 : J-seg -> J-face2 .
  op is-J-face1 : J-seg -> J-face1 .
  op is-F-face1 : F-seg -> F-face1 .
  op is-CF-face2 : C-f-seg -> F-face2 .
  op is-CJ-face2 : C-j-seg -> J-face2 .
  op is-Fixed-Wall : F-face1 End2 F-face2 -> Fixed-Wall .
  op is-Join-Wall : J-face1 End1 J-face2 -> Join-Wall .
  op junc-wall : Fixed-Wall Join-Wall -> Junc-Wall .
  op s-F-face1 : Fixed-Wall -> F-face1 .
  op s-End2 : Fixed-Wall -> End2 .
  op s-F-face2 : Fixed-Wall -> F-face2 .
  op s-J-face1 : Join-Wall -> J-face1 .
  op s-End1 : Join-Wall -> End1 .
  op s-J-face2 : Join-Wall -> J-face2 .
  op s-Fixed-Wall : Junc-Wall -> Fixed-Wall .
  op s-Join-Wall : Junc-Wall -> Join-Wall .
  var F1 : F-face1 . var E2 : End2 . var F2 : F-face2 .
  var J1 : J-face1 . var E1 : End1 . var J2 : J-face2 .
  var FW : Fixed-Wall . var JW : Join-Wall .
  eq s-Fixed-Wall(junc-wall(FW,JW)) = FW .
  eq s-Join-Wall(junc-wall(FW,JW)) = JW .
  eq s-F-face1(is-Fixed-Wall(F1,E2,F2)) = F1 .
  eq s-End2(is-Fixed-Wall(F1,E2,F2)) = E2 .
  eq s-F-face2(is-Fixed-Wall(F1,E2,F2)) = F2 .
  eq s-J-face1(is-Join-Wall(J1,E1,J2)) = J1 .
  eq s-End1(is-Join-Wall(J1,E1,J2)) = E1 .
  eq s-J-face2(is-Join-Wall(J1,E1,J2)) = J2 .
endo
```

D.2.3 A theory of substantiation

```
th SUBSTANCE is protecting JOINING .
  sort  Complex .
```

```
  sorts Object1 Object2 .
  sorts Compnt1 Compnt2 Compnt3 .
  sorts Compnt4 Compnt5 Compnt6 .
  sorts Substance1 Substance2 .
  op declare1 : Substance1 -> Substance1 .
  op declare2 : Substance2 -> Substance2 .
  op assign1 : S-j-seg -> Compnt1 .
  op assign2 : S-f-seg -> Compnt2 .
  op assign3 : M-j-seg -> Compnt3 .
  op assign4 : M-f-seg -> Compnt4 .
  op assign5 : F-seg -> Compnt5 .
  op assign6 : J-seg -> Compnt6 .
  op assign7 : J-seg -> Compnt1 .
  op assign8 : F-seg -> Compnt2 .
  op assign9 : C-f-seg -> Compnt5 .
  op assign10 : C-j-seg -> Compnt6 .
  op comprise1 : Compnt2 Compnt4 Compnt5 -> Object1 .
  op comprise2 : Compnt1 Compnt3 Compnt6 -> Object2 .
  op constitute : Object1 Object2 -> Complex .
  op select-obj1 : Complex -> Object1 .
  op select-obj2 : Complex -> Object2 .
  op select-com2 : Object1 -> Compnt2 .
  op select-com4 : Object1 -> Compnt4 .
  op select-com5 : Object1 -> Compnt5 .
  op select-com1 : Object2 -> Compnt1 .
  op select-com3 : Object2 -> Compnt3 .
  op select-com6 : Object2 -> Compnt6 .
  var O1 : Object1 . var O2 : Object2 .
  var C1 : Compnt1 . var C2 : Compnt2 .
  var C3 : Compnt3 . var C4 : Compnt4 .
  var C5 : Compnt5 . var C6 : Compnt6 .
  eq select-obj1(constitute(O1,O2)) = O1 .
  eq select-obj2(constitute(O1,O2)) = O2 .
  eq select-com2(comprise1(C2,C4,C5)) = C2 .
  eq select-com4(comprise1(C2,C4,C5)) = C4 .
  eq select-com5(comprise1(C2,C4,C5)) = C5 .
  eq select-com1(comprise2(C1,C3,C6)) = C1 .
  eq select-com3(comprise2(C1,C3,C6)) = C3 .
  eq select-com6(comprise2(C1,C3,C6)) = C6 .
endth
```

D.2.4 A parameterised substantiation function

```
obj SUBSTANTIATION[X :: SUBSTANCE] is protecting JOINING .
  sorts Design1 Design2 .
  op substantiate1 : S-Junc Substance1 Substance2 -> Design1 .
  op substantiate2 : C-Junc Substance1 Substance2 -> Design2 .
  op isDesign1 : Substance1 Substance2 Complex -> Design1 .
  op isDesign2 : Substance1 Substance2 Complex -> Design2 .
  vars F1 F2 : Float .
  var S1 : Substance1 . var S2 : Substance2 .
```

```
  var SJUNC : S-Junc . var CJUNC : C-Junc .
  eq substantiate1(SJUNC,S1,S2) = isDesign1(
                  declare1(S1),declare2(S2),constitute(
                  comprise1(assign2(s4-S-Junc(SJUNC)),
                            assign4(s5-S-Junc(SJUNC)),
                            assign5(s6-S-Junc(SJUNC))),
                  comprise2(assign1(s1-S-Junc(SJUNC)),
                            assign3(s2-S-Junc(SJUNC)),
                            assign6(s3-S-Junc(SJUNC))))) .

  eq substantiate2(CJUNC,S1,S2) = isDesign2(
                  declare1(S1),declare2(S2),constitute(
                  comprise1(assign8(s4-C-Junc(CJUNC)),
                            assign4(s5-C-Junc(CJUNC)),
                            assign9(s6-C-Junc(CJUNC))),
                  comprise2(assign7(s1-C-Junc(CJUNC)),
                            assign3(s2-C-Junc(CJUNC)),
                            assign10(s3-C-Junc(CJUNC))))) .
endo
```

D.2.5 A view from SUBSTANCE to WALL

```
view WALLV from SUBSTANCE to WALL is
  sort Substance1 to Wall-Type .
  sort Substance2 to Material .
  sort Compnt1 to J-face1 .
  sort Compnt2 to F-face1 .
  sort Compnt3 to End1 .
  sort Compnt4 to End2 .
  sort Compnt5 to F-face2 .
  sort Compnt6 to J-face2 .
  sort Object1 to Fixed-Wall .
  sort Object2 to Join-Wall .
  sort Complex to Junc-Wall .
  op declare1 to is-wall-type-of .
  op declare2 to is-material-of .
  op assign1 to is-SJ-face1 .
  op assign2 to is-SF-face1 .
  op assign3 to is-End1 .
  op assign4 to is-End2 .
  op assign5 to is-F-face2 .
  op assign6 to is-J-face2 .
  op assign7 to is-J-face1 .
  op assign8 to is-F-face1 .
  op assign9 to is-CF-face2 .
  op assign10 to is-CJ-face2 .
  op comprise1 to is-Fixed-Wall .
  op comprise2 to is-Join-Wall .
  op constitute to junc-wall .
  op select-obj1 to s-Fixed-Wall .
  op select-obj2 to s-Join-Wall .
```

```
  op select-com1 to s-J-face1 .
  op select-com2 to s-F-face1 .
  op select-com3 to s-End1 .
  op select-com4 to s-End2 .
  op select-com5 to s-F-face2 .
  op select-com6 to s-J-face2 .
endv
```

D.3 Wall Junction Design: A Case of Joint Substantiation

D.3.1 Derivative operations

```
obj DERIVING is protecting JOINING .
  sorts SJ-DS1 SJ-DS2 CJ-DS1 CJ-DS2 .
  op derive1 : S-Junc -> SJ-DS1 .
  op derive2 : S-Junc -> SJ-DS2 .
  op derive3 : C-Junc -> CJ-DS1 .
  op derive4 : C-Junc -> CJ-DS2 .
  op isSJ-DS2 : S-j-seg M-j-seg J-seg -> SJ-DS2 .
  op isSJ-DS1 : S-f-seg M-f-seg F-seg -> SJ-DS1 .
  op isCJ-DS2 : J-seg M-j-seg C-j-seg -> CJ-DS2 .
  op isCJ-DS1 : F-seg M-f-seg C-f-seg -> CJ-DS1 .
  op s-S-j-seg : SJ-DS2 -> S-j-seg .
  op s-M-j-seg : SJ-DS2 -> M-j-seg .
  op s-J-seg : SJ-DS2 -> J-seg .
  op s-S-f-seg : SJ-DS1 -> S-f-seg .
  op s-M-f-seg : SJ-DS1 -> M-f-seg .
  op s-F-seg : SJ-DS1 -> F-seg .
  op s-J-seg : CJ-DS2 -> J-seg .
  op s-M-j-seg : CJ-DS2 -> M-j-seg .
  op s-C-j-seg : CJ-DS2 -> C-j-seg .
  op s-F-seg : CJ-DS1 -> F-seg .
  op s-M-f-seg : CJ-DS1 -> M-f-seg .
  op s-C-f-seg : CJ-DS1 -> C-f-seg .

  var SJUNC : S-Junc . var CJUNC : C-Junc .
  var SJ : S-j-seg . var MJ : M-j-seg . var J : J-seg .
  var SF : S-f-seg . var MF : M-f-seg . var F : F-seg .
  var CJ : C-j-seg . var CF : C-f-seg .
  eq derive2(SJUNC) = isSJ-DS2(s1-S-Junc(SJUNC),s2-S-Junc(SJUNC),
                               s3-S-Junc(SJUNC)) .
  eq derive1(SJUNC) = isSJ-DS1(s4-S-Junc(SJUNC),s5-S-Junc(SJUNC),
                               s6-S-Junc(SJUNC)) .
  eq derive4(CJUNC) = isCJ-DS2(s1-C-Junc(CJUNC),s2-C-Junc(CJUNC),
                               s3-C-Junc(CJUNC)) .
  eq derive3(CJUNC) = isCJ-DS1(s4-C-Junc(CJUNC),s5-C-Junc(CJUNC),
                               s6-C-Junc(CJUNC)) .
  var SJDS1 : SJ-DS1 . var SJDS2 : SJ-DS2 .
  var CJDS1 : CJ-DS1 . var CJDS2 : CJ-DS2 .
```

```
  eq s-S-j-seg(isSJ-DS2(SJ,MJ,J)) = SJ .
  eq s-M-j-seg(isSJ-DS2(SJ,MJ,J)) = MJ .
  eq s-J-seg(isSJ-DS2(SJ,MJ,J)) = J .
  eq s-S-f-seg(isSJ-DS1(SF,MF,F)) = SF .
  eq s-M-f-seg(isSJ-DS1(SF,MF,F)) = MF .
  eq s-F-seg(isSJ-DS1(SF,MF,F)) = F .
  eq s-J-seg(isCJ-DS2(J,MJ,CJ)) = J .
  eq s-M-j-seg(isCJ-DS2(J,MJ,CJ)) = MJ .
  eq s-C-j-seg(isCJ-DS2(J,MJ,CJ)) = CJ .
  eq s-F-seg(isCJ-DS1(F,MF,CF)) = F .
  eq s-M-f-seg(isCJ-DS1(F,MF,CF)) = MF .
  eq s-C-f-seg(isCJ-DS1(F,MF,CF)) = CF .
endo
```

D.3.2 A parameterised function for joint substantiation

```
obj JOINT-SUBSTAN[X :: SUBSTANCE] is protecting DERIVING .
  sorts  Domain-A-Exp Domain-B-Exp .
  op substantiate-a1 : SJ-DS1 Substance1 Substance2 -> Domain-A-Exp .
  op substantiate-b1 : SJ-DS2 Substance1 Substance2 -> Domain-B-Exp .
  op substantiate-a2 : CJ-DS1 Substance1 Substance2 -> Domain-A-Exp .
  op substantiate-b2 : CJ-DS2 Substance1 Substance2 -> Domain-B-Exp .
  op isDDEa : Substance1 Substance2 Object1 -> Domain-A-Exp .
  op isDDEb : Substance1 Substance2 Object2 -> Domain-B-Exp .
  var S1 : Substance1 . var S2 : Substance2 .
  var SJDS1 : SJ-DS1 . var SJDS2 : SJ-DS2 .
  var CJDS1 : CJ-DS1 . var CJDS2 : CJ-DS2 .
  eq substantiate-b1(SJDS2,S1,S2) = isDDEb(
                    declare1(S1),declare2(S2),
                    comprise2(assign1(s-S-j-seg(SJDS2)),
                              assign3(s-M-j-seg(SJDS2)),
                              assign6(s-J-seg(SJDS2)))) .

  eq substantiate-a1(SJDS1,S1,S2) = isDDEa(
                    declare1(S1),declare2(S2),
                    comprise1(assign2(s-S-f-seg(SJDS1)),
                              assign4(s-M-f-seg(SJDS1)),
                              assign5(s-F-seg(SJDS1)))) .

  eq substantiate-b2(CJDS2,S1,S2) = isDDEb(
                   declare1(S1),declare2(S2),
                   comprise2(assign7(s-J-seg(CJDS2)),
                             assign3(s-M-j-seg(CJDS2)),
                             assign10(s-C-j-seg(CJDS2)))) .

  eq substantiate-a2(CJDS1,S1,S2) = isDDEa(
                   declare1(S1),declare2(S2),
                   comprise1(assign8(s-F-seg(CJDS1)),
                             assign4(s-M-f-seg(CJDS1)),
                             assign9(s-C-f-seg(CJDS1)))) .
endo
```

D.3.3 Designer A's world for modelling Wall-A

```
obj WALL-A is protecting JOINING .
  sort Hide .    sort Wall-A .    sorts Material Install .
  sorts F-face1 End2 F-face2 .
  op brick : -> Material . op block : -> Material .
  op concrete : -> Material . op window-1 : -> Install .
  op window-2 : -> Install . op window-3 : -> Install .
  op is-SF-face1 : S-f-seg -> F-face1 .
  op is-End2 : M-f-seg -> End2 .
  op is-F-face2 : F-seg -> F-face2 .
  op is-F-face1 : F-seg -> F-face1 .
  op is-CF-face2 : C-f-seg -> F-face2 .
  op is-Wall-A : F-face1 End2 F-face2 -> Wall-A .
  op install : Install -> Install .
  op is-material-of : Material -> Material .
  op s-F-face1 : Wall-A -> F-face1 .
  op s-End2 : Wall-A -> End2 .
  op s-F-face2 : Wall-A -> F-face2 .    op hide : Hide -> Hide .
  var F1 : F-face1 . var E : End2 . var F2 : F-face2 .
  eq s-F-face1(is-Wall-A(F1,E,F2)) = F1 .
  eq s-End2(is-Wall-A(F1,E,F2)) = E .
  eq s-F-face2(is-Wall-A(F1,E,F2)) = F2 .
endo
```

D.3.4 Designer B's world for modelling Wall-B

```
obj WALL-B is protecting JOINING .
  sort Hide .    sort Wall-B .    sorts Cladding Fenestrate .
  sorts J-face1 End1 J-face2 .
  op ext-cladding : -> Cladding .  op int-cladding : -> Cladding .
  op window-a-door-1-window-b : -> Fenestrate .
  op window-c-door-2-window-d : -> Fenestrate .
  op is-SJ-face1 : S-j-seg -> J-face1 .
  op is-End1 : M-j-seg -> End1 .
  op is-J-face2 : J-seg -> J-face2 .
  op is-J-face1 : J-seg -> J-face1 .
  op is-CJ-face2 : C-j-seg -> J-face2 .
  op is-Wall-B : J-face1 End1 J-face2 -> Wall-B .
  op is-cladded-with : Cladding -> Cladding .
  op fenestrate : Fenestrate -> Fenestrate .
  op s-J-face1 : Wall-B -> J-face1 .
  op s-End1 : Wall-B -> End1 .
  op s-J-face2 : Wall-B -> J-face2 .      op hide : Hide -> Hide .
  var J1 : J-face1 . var E : End1 . var J2 : J-face2 .
  eq s-J-face1(is-Wall-B(J1,E,J2)) = J1 .
  eq s-End1(is-Wall-B(J1,E,J2)) = E .
  eq s-J-face2(is-Wall-B(J1,E,J2)) = J2 .
endo
```

Appendix E

Published Papers of the Thesis

E.1 Exploring Communication in Collaborative Design: Cooperative Architectural Modelling

Published in the Journal of *Design Studies*, Vol.15, No.1, pages 19–44, Oxford: Butterworth-Heinemann Ltd, January 1994.

E.2 Survey of Collaborative Drawing Support Tools: Design Perspectives and Prototypes.

Published in the Journal of *Computer Supported Cooperative Work*, Vol. 1, No. 3, pages 197–228, Dordrecht: Kluwer Academic Publishers, October 1993.

E.3 On the Emergence of Common Design Metaphors in Collaborative Design

A paper accepted for *AAAI 1993 Fall Syposium Series, "Human-Computer Collaboration: Reconciling Theory, Synthesizing Practice"*, October 22-24, 1993. Research Triangle Park, Raleigh, North Carolina, US., included in the Symposium's Technical Report (FS-93-05) pages 69–74, AAAI Press.

E.4 Supporting Heterogeneous Distributed Substantiation of Common Generic Structures in Collaborative Design

A paper presented in *AAAI-93 Workshop on Artificial Intelligence in Collaborative Design*, 11 July 1993, Washington D.C., included in the Workshop Working Notes, pages 149–170.

E.5 Participatory Architectural Modelling: Common Images and Distributed Design Developments

Published in the *Proceedings of Participatory Design Conference (PDC'92)*, pages 171–180, Computer Professional Social Resposibility (CPSR). Massachusetts Institute of Technology, Cambridge MA, US., 6–7 November 1992.

E.6 A Formal Perspective on Teamwork in Design Modelling

Published in *Edinburgh Architecture Research*, Vol. 19, pages 137–154, Department of Architecture, University of Edinburgh, April 1992. (An adapted version of this paper was accepted by the HCI'92 Conference Committee for a poster presentation at the University of York, 15–18 September 1992.)

To abide by the University thesis submission regulation, copies of the published papers are sewn in at the back of the thesis in the above order.

Bibliography

[Ale64] C. Alexander. *Notes on the Synthesis of Form.* Cambridge : Harvard University Press, 1964.

[Ban60] R. Banham. *Theory and Design in the First Machine Age.* The Architectural Press London, 1960.

[Bar89] J. Barwise. *The Situation in Logic.* CSLI Lecture Notes No. 17, Center for the Study of Language and Information, Stanford, 1989.

[BBP89] B. Banieqbal, H. Barringer, and A. Pnueli. Temporal logic in specification. In *Proceedings of the Colloquium on Temporal Logic in Specification.* Berlin London: Springer-Verlag, 1989.

[BBS75] T. Benton, C. Benton, and D. Sharp. *Form and Function: A Source Book for the History of Architecture and design 1890–1939.* Crosby Lockwood Staples London in association with the Open University Press, 1975.

[BG92] T. Brinck and L. M. Gomez. A collaborative medium for the support of conversational props. In *Proceedings of CSCW'92,* pages 171–178. ACM Press, 1992.

[BHK89] J. A. Bergstra, J. Heering, and P. Klint. *Algebraic Specification.* New York: ACM Press, 1989.

[Bij86] A. Bijl. Designing with words and pictures in a logic modelling environment. In *Computer-aided Architectural Design Futures,* pages 23–29. International Conference on Computer-aided Architectural Design. Butterworths, Guildford, UK, 1986.

[Bij87] A. Bijl. Strategies for CAD. In *Intelligent CAD Systems 1: Theoretical and Methodological Aspects,* pages 2–19. Berlin: Springer-Verlag, 1987.

[Bij89] A. Bijl. *Computer Discipline and Design Practice.* Edinburgh University Press, 1989.

[BLN86] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodilogies for database schema integration. *ACM Computing Survey,* 18(4):323–364, 1986.

[Bly88] S. L. Bly. A use of drawing surfaces in different collaborative settings. In I. Greif, editor, *Proceedings of the Conference on Computer- Supported Cooperative Work (CSCW'88),* pages 250–256. ACM Press, 1988.

[BM89] S. L. Bly and S. L. Minneman. Commune: A shared drawing surface. Technical Report SSL-89-86, System Sciences Laboratory, Palo Alto Research Center, 1989.

[Bon89] A. H. Bond. The cooperation of experts in engineering design. In Les Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence Vol. 2 (Research Notes in Artificial Intelligence),* pages 463–483. London: Pitman, 1989.

[BP83] J. Barwise and J. Perry. *Situations and Attitudes.* The MIT Press, 1983.

[BR92] A.H. Bond and R.J. Ricci. Cooperation in aircraft design. *Research in Engineering Design*, 4(4):115–130, 1992.

[BS91] L. J. Bannon and K. Schmidt. CSCW: Four characters in search of a context. In *Studies in Computer Supported Cooperative Work: Theory, Practice and Design*, pages 3–16. North-Holland: Elsevier Science Publishing Company, Inc, 1991.

[BSR79] A. Bijl, D. Stone, and D. Rosenthal. Integrated caad systems. Technical report, EdCAAD Studies, Department of Architecture, University of Edinburgh, 1979.

[BST89] H. E. Bal, J. G. Steiner, and A. S. Tanenbaum. Programming languages for distributed computing systems. *ACM Computing Survey*, 21(3):261–322, Sep. 1989.

[Buc88] L. Bucciarelli. An ethnographic perspective on engineering design. *Design Studies*, 9(3):159–168, July 1988.

[CA68] D. Cartwright and Zander A. *Group Dynamics : Research and Theory.* London : Tavistock Publications, 1968. Third ed. originally published, New York: Harper & Row, 1968. Previous ed., Evanston: Row, Peterson, 1960; (B61-1357), London: Tavistock Publications, 1961.

[CB88] J Conklin and M. L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transaction on Office Information Systsems*, 6(4):303–331, October 1988.

[CN83] G. R. Collins and J. B. Nonell. *The Designs and Drawings of Antonio Gaudí*, pages 31–35. Princeton, N.J. Guildford: Princeton University Press, 1983.

[Cra87] C. Crampton. MUSK - a multi-user sketch program. In *Proceedings of the European UNIX Systems User Group*, pages 17–29, 1987.

[Cra91] I. D. Craig. *Formal Specification of Advanced AI Architectures.* Ellis Horwood, 1991.

[DC91a] P. Dewan and R. Choudhary. Flexible user interface coupling in a collaborative system. In *Proceedings of the ACM CHI'91 Conference*, pages 41–48. ACM Press, 1991.

[DC91b] P. Dewan and R. Choudhary. Primitives for programming multi-user interfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'91)*, pages 69–78. ACM Press, 1991.

[Dev91] K. Devlin. *Information and Logic.* Cambridge University Press, 1991.

[Dev92a] K. Devlin. Situation theory and social structure. In *Proceedings of the Conference on Applied Logic.* Springer-Verlag, 1992. Amsterdam, December 1992.

[Dev92b] K. Devlin. Situation theory and the design of interactive information systems. Technical Report CSLI-92-171, Center for the Study of Language and Information, July 1992.

[DR93] K. Devlin and D. Rosenberg. Situation theory and cooperative action. In *Situation Theory and its Applications (Volume 3).* CSLI, Stanford, 1993. (To appear in CSLI Lecture Note Series).

[Dre81] F. I. Dretske. *Knowledge and the Flow of Information.* Oxford: Blackwell, 1981.

[Dur75] J. N. L. Durand. *Precis des lecons d'architecture donnees a l'Ecole royale polytechnique.* Unterschneidheim, Ger.: Uhl, 1975. Reprint of the 1819 ed., published by the author, Paris, issued with the author's *Partie graphique des cours d'architecture.*

[EGR91] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: Some issues and experiences. *Communcations of the ACM*, 34(1):39–58, 1991.

[FGL+92] G. Fischer, J. Grudin, A. Lemke, R. McCall, J. Ostwald, B. Reeves, and F. Shipman. Supporting indirect, collaborative design with integrated knowledge-based design environments. *Human Computer Interaction*, 7(3):281–314, 1992.

[FKN+92] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: A framework for integrating multiple perspectives in systems development. *International Journal of Software Engineering & Knowledge Engineering*, 1992. To appear.

[FLMI91] G. Fischer, A. Lemke, T. Mastaglio, and Morch A. I. The role of critiquing in cooperative problem solving. *ACM Transactions on Information Systems*, 9(3):123–151, April 1991.

[Ge88] I. Greif (ed.). *Computer-Supported Cooperative Work: A Book of Readings.* San Mateo, California: Morgan Kaufmann, 1988.

[GEAF88] M.D. Gross, S.M. Ervin, J.A. Anderson, and A. Fleisher. Constraints: Knowledge representation in design. *Design Studies*, 9(3):133–143, July 1988.

[Gog89] J. A. Goguen. Modular algebraic specification of some basic geometrical connstructions. In D. Kapur and J. L. Mundy, editors, *Geometric Reasoning*, pages 123–153. Elsevier Science Publishers B.V., 1989.

[Gol88] G. Goldschmidt. Interpretation: its role in architectural design. *Design Studies*, 9(4):235–245, 1988.

[Gre91] S. Greenberg. An annotated bibliography of Computer Supported Cooperative Work. *SIGCHI Bulletin*, 23(3):29–62, July 1991.

[GRHL89] Les Gasser, N. F. Rouquette, R. W. Hill, and J. Lieb. Representing and using organizational knowledge in distributed ai systems. In Les Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence Vol. 2 (Research Notes in Artificial Intelligence)*, pages 55–78. London: Pitman, 1989.

[Gru91] J. Grudin. Introduction to the special issue on cscw. *Communications of the ACM*, 34(12):30–34, December 1991.

[GRWB91] S. Greenberg, M. Roseman, D. Webster, and R. Bohnet. Issues and experiences designing and implementing two group drawing tools. Technical Report 91/438/22, Department of Computer Science, University of Calgary, July 1991.

[GW88] J. A. Goguen and T. Winkler. Introducing OBJ3. Technical Report SRI-CSL-88-9, SRI International, Computer Science Laboratory, August 1988.

[Hal69] L. Halprin. *The RSVP Cycles: Creative Processes in the Human Environment*, pages 54–57. George Braziller Inc., New York, 1969.

[Har89] R. Harrison. *Abstract Data Type in Modula-2.* Chichester: John Wiley & Sons Ltd., 1989.

[HG88] N.J. Habraken and M.D. Gross. Concepts design games. *Design Studies*, 9(3):150–158, July 1988.

[HKLP92] A. Hars, K. Kosanke, J. Langemeyer, and C. Petrie. The Model/Application Link. In *Proceedings of the First International Conference on Enterprise Integration Modelling*, pages 42–46. The MIT Press, 1992.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Englewood Cliffs London : Prentice Hall, 1985.

[Hos77] E. M. Hoskins. The oxsys system. In J. S. Gero, editor, *Computer Applications in Architecture*, pages 343–391. Applied Science, 1977.

[IK92] H. Ishii and M. Kobayashi. ClearBoard: A seamless medium for shared drawing and conversation with eye contact. In *Proceedings of CHI'92*, pages 525–532. ACM Press, 1992.

[IKG92] H. Ishii, M. Kobayashi, and J. Grudin. Integration of inter-personal space and shared workspace: ClearBoard design and experiments. In J. Turner and R. Kraut, editors, *Proceedings of CSCW'92*, pages 33–42. ACM Press, 1992.

[IM91] H. Ishii and N. Miyake. Toward an open shared workspace: computer and video fusion approach of TeamWorkStation. *Communications of the ACM*, 34(12):37–50, December 1991.

[IO90] H. Ishii and M. Ohkubo. Design of TeamWorkStation: A realtime shared workspace fusing desktops and computer screen. In S. Gibbs and A. A. Verrighn-Stuart, editors, *Proceedings of IFIP WG8.4 Conference on Multi- User Interfaces and Applications.* North Holland, 1990.

[Jan78] R. Janke. *Architectural Models.* London Academy Editions, 1978.

[Joh88] R. Johansen. *Groupware: Computer Support for Business Team.* The Free Press, N. Y., 1988.

[Jol73] J. L. Jolley. *The Fabric of Konwledge.* Gerald Duckworth & Co. Ltd., London, 1973.

[Jon86] C. B. Jones. *Systematic Software Development Using VDM.* New York London: Prentice Hall, 1986.

[Kah35] E. J. Kahn. *Design in Art and Industry.* New York: C. Scribner's Sons, 1935.

[KR70] W. Kunz and H. W. J. Rittel. Issues as elements of information systems. Technical Report 131, Institute of Urban and Regional Development, University of California, Berkeley, California, 1970.

[Kri85] R. Krishnamurti. A model for design description. *Edinburgh Architecture Research*, 12:71–89, 1985.

[Lak80] G. Lakoff. *Metaphors We Live By.* Chicago: University of Chicago Press, 1980.

[Lak83] F. Lakin. Measuring text-graphic activity. In *Proceedings of the GRAPHICS INTERFACE'83*, 1983. Edmonton, Alberta.

[Lak86] F. Lakin. Spatial parsing for visual languages. In S. K. Chang, T. Ichikawa, and Ligomenides P. A., editors, *Visual languages*, pages 35–85. Plenum Press, 1986.

[Lak88] F. Lakin. A performing medium for working group graphics. In I. Greif, editor, *Computer Supported Cooperative Work: A Book of Readings*, pages 366–396. Morgan Kaufman Publishers, 1988.

[Lak90] F. Lakin. Visual languages for cooperation: A performing medium approach to systems for cooperative work. In J. Galegher, R. E. Kraut, and C. Egido, editors, *Intellectual Teamwork: Social and technological Foundations of Cooperative Work*, pages 453–488. Lawrence Erlbaum Associates, 1990.

[Lam77] W. M. C. Lam. *Perception and Lighting as Formgivers for Architecture*, pages 125–129. McGraw-Hill Book Company, 1977.

[Law80] B Lawson. *How Designers Think*, pages 171–186. The Architectural Press Ltd: London, 1980.

[LM91] I. M. Lu and M. Mantei. Idea management in a shared drawing tool. In L. Bannon, M. Robinson, and Schmit K., editors, *Proceedings of ECSCW'91*, pages 97–112. Kluwer Academic Publisher, 1991.

[Lu92] I. M. Lu. Supporting Idea Management in a Shared Drawing Tool. Unpublished Master Thesis, Department of Computer Science, University of Toronto, 1992.

[Mar79] C. Martinell. *Gaudí: his Life, his Theories, his Work*, page 335. Barcelona Editorial Blume, 1979.

[Mid67] M. Middleton. *Group Practice in Design*. London: The Architectural Press, 1967.

[Mil89] R. Milner. *Communication and Concurrency*. New York London : Prentice Hall, 1989.

[MMS92] C. Menzel, R. J. Mayer, and L. K. Sanders. Representation, information flow, and model integration. In *Proceedings of the First International Conference on Enterprise Integration Modeling*, pages 131–141. Cambridge, Massachusetts: The MIT Press, 1992.

[MN91] T. Mori and H. Nakagawa. A formalization of metaphor understanding in situation semantics. In *Situation Theory and Its Applications (Volume 2)*, pages 449–468. CSLI, Stanford, 1991. CSLI Lecture Notes No. 26.

[oD80] United States Department of Defense. *Reference Manual for the ADA Programming Language*. DOD Management Steering Committe for Embedded Computer Resources, 1980.

[Pat91] J. F. Patterson. Comparing the programming demands of single-user and multi-user applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'91)*, pages 87–94, 1991.

[PB88] J.C. Platt and A.H. Barr. Constraint methods for flexible models. *ACM Computer Graphics*, 22(4):279–288, 1988.

[Pen89] C. Peng. Knowledge and Graphics: A Study of the Approaches to Knowledge Based CAD. Unpublished Master Thesis, Department of Architecture and Building Science, University of Strathclyde, 1989.

[PST91] B. Potter, J. Sinclair, and B. Till. *An Introduction to Formal Specification and Z*. Prentice Hall, 1991.

[Reh85] D.R. Rehak. Interfacing expert systems with design databases in integrated cad systems. *CAD*, 19(9):235–245, November 1985.

[Row87] P. G. Rowe. *Design Thinking*, pages 1–39. The MIT Press, 1987.

[RS92] B. Reeves and F. Shipman. Supporting communication between designers with artifact-centered evolving information spaces. In J. Turner and R. Kraut, editors, *Proceedings of CSCW'92*, pages 394–401. ACM Press, 1992.

[SB92] K. Schmidt and L. Bannon. Taking CSCW seriously: Supporting articulation work. *Computer Supported Cooperative Work*, 1(1–2):7–40, 1992.

[Sch80] D. L. Schodek. *Structures.* London: Prentice-Hall, 1980.

[Sch85] D. Schön. *The Design Studio: An Exploration of its Traditions and Potentials*, pages 30–52. RIBA Publications Ltd., 1985.

[Sch90] G. Schmitt. IBDE, VIKA, ARCHPLAN: Architectures for design knowledge representation, acquisition and application. In H. Yoshikawa and T. Holden, editors, *Intelligent CAD, II.* North-Holland, 1990.

[Sch91] D. Schön. *The Reflective Practitioner: How Professionals Think in Action*, pages 129–168. Aldershot Hants: Avebury, 1991.

[SFB+87] M. Stefik, G. Foster, D. Bobrow, K. Kahn, S. Lanning, and L. Suchman. Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *Communications of the ACM*, 30(1):32–47, 1987.

[Sil92] B. G. Silverman. Survey of expert critiquing systems: Practical and theoretical frontiers. *Communications of the ACM*, 35(4):106–127, April 1992.

[SK87] M. Sloman and J. Kramer. *Distributed Systems and Computer Network*, pages 128–129. Prentice-Hall, 1987.

[Smi79] R. G. Smith. *A Framework for Distributed Problem Solving.* Ann Arbor, Michigan: UMI Research Press, 1979.

[Spe91] M. Speidel. *Team Zoo.* London: Thames and Hudson Ltd, 1991. Translated from the German by Michael Robinson. Translated from the Japanese by Manfred Speidel.

[Spi89] J. M. Spivey. *The Z Notation: A Reference Manual.* New York London: Prentice Hall, 1989.

[Sta89] S. L. Star. The structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving. In Les Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence Vol. 2 (Research Notes in Artificial Intelligence)*, pages 37–53. London: Pitman, 1989.

[Sti92] C. Stirling. Modal and temporal logics for proceses. Technical Report ECS-LFCS-92-221, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1992.

[Tan89] J. C. Tang. Listing, drawing, and gesturing in design: A study of the use of shared workspace by design teams. Technical Report SSL-89-3, Palo Alto Research Center, 1989.

[Tan91] J. C. Tang. Findings from observational studies of collaborative work. *International Journal of Man Machine Studies*, 34:143–60, 1991.

[TAS79] E. Tjalve, M. M. Andereasen, and F. Frackmann Schmidt. *Engineering Graphic Modelling.* Newnes-Butterworth, 1979.

[TB88] C. Tweed and A. Bijl. MOLE: A reasonable logic for design. *Edinburgh Architecture Research*, 15:106–140, 1988.

[TL87] A. Tzonis and L. Lefaivre. *Classical Architecture: The Poetics of Order*, pages 9–25. Cambridge, Mass. London: MIT Press, 1987.

[TL88] J. C. Tang and L. J. Leifer. A framework for understanding the workspace activity of design teams. In *Proceedings of the Conference on Computer- Supported Cooperative Work (CSCW'88)*, pages 26–28. ACM Press, 1988.

[TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *ACM Computer Graphics*, 21(4):205–214, July 1987.

[Ver91] S. Vergopoulos. A multi-layered model for interpreting design drawings. In *Proceedings of the EuropIA'91*, pages 150–167, 1991.

[War87] T. Ward. Design archetypes from group processes. *Design Studies*, 8(3):157–169, 1987.

[WFB87] A. Witkin, K. Fleischer, and A. Barr. Energy constraints on parameterized models. *ACM Computer Graphics*, 21(4):225–232, July 1987.

[Whi92] R. White. FireMOLE – an architecture for reasoning with drawings. *Design Studies*, 13(3):320–334, 1992.

[Wil91] P. Wilson. *Introduction to Computer Supported Cooperative Work.* Oxford Intellect, 1991.

[WL88] J. Woodcock and M. Loomes. *Software Engineering Mathematics.* Pitman, 1988.

[WP87] F. Woods and J. Powell. *Overlay Drafting: A Primer for the Building Design Team.* London: Architectural Press, 1987.

[WP92] C. Willis and D. Paddon. *Abstraction & Specification with Modula-2.* London: Pitman Publishing, 1992.

Journal Offprint Paper

BUTTERWORTH HEINEMANN

A member of the Reed Elsevier group

Linacre House, Jordan Hill, Oxford OX2 8DP, UK

OFFPRINT PAPERS

Butterworth-Heinemann are leading international publishers of a range of highly respected academic journals in Applied Science, Information Technology, Engineering, Materials Science, Policy Studies, Social Sciences, Biomedical Sciences and Medicine. Our journals are renowned for their high quality of both content and presentation. Papers are peer-reviewed and all our journals have distinguished editorial boards.

If you would like to receive subscription details, a sample copy or other information about the journal in which this offprint appeared, please complete and return the form provided below. Information about our other titles is available from our price list or catalogue (see form below).

Notes for Authors

To obtain details on how to submit papers to one of our journals, please contact:
The Group Editor at the Oxford address below (please state the appropriate journal title).

Reprints

We can reprint any article appearing in our journals and tailor it to your needs with an advert or logo. For a quotation on your reprint requirements, please contact:
The Reprints Department at the Oxford address below.

For further information about any other aspect of our journals, please contact us at:
Butterworth-Heinemann Ltd., Linacre House, Jordan Hill, Oxford, OX2 8DP, UK
Tel: +44 (0) 865 310366 **Fax: +44 (0) 865 310898**

✂ Please cut here

Return Form

Area(s) of Interest: .....

Journal Title(s) .....

.....

☐ ***Subscription Details Request***
Please send me full subscription details and information about the above journal title(s)

☐ ***Sample Copy Request Form***
Please send me a free sample copy of the above journal title(s)

☐ ***Journals Price List***
Please send me a full price list for all your journals

☐ ***Catalogue***
Please send me a catalogue of your journals

Name .....

Organisation .....

Address .....

.....

Postcode / Zipcode ..... Country .....

Telephone ..... E-mail .....

Fax ..... Signature ..... Date .....

Please return this form to:

Butterworth-Heinemann Ltd.
Turpin Distribution Services Ltd.
Blackhorse Road
Letchworth
Herts, SG6 1HN, UK
Tel: +44 (0) 462 672555
Fax: +44 (0) 462 480947

OR

Journal Fulfilment Department
Butterworth-Heinemann US
80 Montvale Avenue
Stoneham
MA 02180, USA
Tel: +1 617 438 8464
Fax: +1 617 438 1479

Exploring communication in collaborative design: co-operative architectural modelling

Chengzhi Peng, Edinburgh Computer-Aided Architectural Design, Department of Architecture, University of Edinburgh, 20 Chambers Street, Edinburgh EH1 1JZ, UK

An exploration of communication in collaborative design from the perspective of co-operative architectural modelling is reported. The objectives and problems of communication in collaborative design are described and analysed by viewing design as, basically, disciplines of modelling complex objects. Three cases of teamwork in architectural modelling are studied, each demonstrating a rich and informative approach to collaboration. Looking at the cases from the co-operative modelling perspective, important conditions for communication are observed: firstly, the participation and co-ordination among heterogeneous systems of representation and action that individual members of a design team work with; and secondly, the interconnection between common goals shared by all participants and domain-oriented goals pursued by individuals. In exploring how the conditions were met, it was found useful to characterize communication in terms of the inter-relations between common images and distributed design developments. Two generic patterns of communication in collaborative design were found, which suggest two alternative conceptual frameworks for developing computational representations.

Keywords: collaborative design, architectural modelling, distributed design developments, computer-supported collaborative design

Like many other human activities, the design and construction of built environments always involves many designers working jointly as teams. There are two main reasons why a study of collaborative design in buildings, or, indeed, in any other kinds of complex artifact, presents an intrinsic research interest, firstly, for technical necessities, participants work with individual object worlds (Note 1), calling upon heterogeneous conceptual structures and instruments; and secondly for critical judgements, the emergence of final unity in design products as a whole is of essential common concern shared by all parties in a design team, which is closely connected with the developments of design solutions pursued in the individual worlds (Note 2). Without a conceptual understanding of how these basic conditions are met, an

1 Bucciarelli, L 'An ethnographic perspective on engineering design', *Design Studies* Vol 9 No 3 (1988) 159–168

2 Bucciarelli, L, Goldschmidt, G and Schön, D 'Generic design process in architecture and engineering', in J P Protzen, (Ed), *Proceedings of the 1987 Conference on Planning and Design in Architecture*, Boston, MA, August 1978, pp 59–64

3 Kahn, E J *Design in Art and Industry* C. Scribner, New York 1935.

4 Middleton, M *Group Practice in Design*. The Architectural Press, London (1967) p 279

0142–694X/94/010019–26

ontological account of what communication or computation tools are needed for collaborative design work cannot be reached. To contribute to such an understanding, this paper aims to discuss communication in collaborative design by examining some teamwork approaches to building design. The reults are represented by alternative conceptual frameworks that serve as references in the further development of computational representations and a definition of collaborative design environments.

Certainly, the phenomenon of a group approach to design is not an entirely new subject within design studies. A variety of interpretative frameworks have been proposed. One widely taken metaphor of group design process is, perhaps, design as *game*. Lawson[5] reviewed several design games that were specially set up to model group dynamics in architectural and urban design. Working in line with the game metaphor, Habraken and Gross[6] invented a computer program called 'concept design game' which can record and then replay sessions of participants' playing for control distribution and territorial organization. By conceptualizing one of his protocol analyses of a design dialogue between an architecture student and a studio master, Schön[7,8] proposed the theory of 'reflection-in-action', acting as an epistemological framework of design learning. To add just a further example, a studio-based empirical study of a design project participated by a group of architecture students was carried out by Ward[9], in which seven subjects went through group processes and developed *archetypes* for a commercial complex project mainly by gathering together individually made cardboard models.

1 *Collaborative design*

Recently, collaborative design has been studied by some computer scientists working in the field of computer-supported co-operative work (CSCW). The problem of what and how to develop communication or computer systems that can support people involved in design teamwork has become an active research area. In a recent bibliographical survey, Greenberg[10] introduced the keyphrase 'shared workspaces' as a distinct subarea within CSCW. This survey shows that a large portion of the research work on shared workspaces has to do with the understanding and construction of 'shared drawing spaces'. Work on shared drawing/design systems is exploring novel dimensions, such as human–computer interaction situated in a group context and the structures of distributed graphics etc., which are not normally seen in traditional computer-aided design or drawing systems. Unlike such conventional systems, CSCW-oriented designs have attempted to provide *tele-presence* or *tele-data*, facilitating direct or indirect communications among members of a design team (Note 3).

5 **Lawson, B** *How Designers Think*, Architectural Press, London (1980) pp 171–186

6 **Habraken, N and Gross, M** 'Concepts design games', *Design Studies*, 9(3):150–158, July 1988

7 **Schön, D** *The Reflective Practitioner: How Professionals Think in Action*, Avebury, Aldershot, UK (1991), pp 129–168

8 **Schön, D** *The Design Studio: An Exploration of its Traditions and Potentials* RIBA, London (1985) pp 30–32

9 **Ward, T** 'Design archetypes from group processes', *Design Studies*, Vol 8 No 3 (1987) 157–169

10 **Greenberg, S** 'An annotated bibliography of computer supported cooperative work', *SIGCHI Bulletin*, Vol 23 No 3 (1991) 29–62

However, due to the richness and complexity of collaborative design activity, some problematic situations remain unexplored. Especially, little is known about the nature of co-operation where participants of different technical specializations communicate and co-ordinate with each other to achieve, or, to cope with, *design unity* in final products. This paper suggests that the communication of that nature in collaborative design can be better understood by examining what is involved when designers work jointly in *modelling* the design of complex objects like buildings. It is believed that, by looking at *co-operative architectural modelling*, a clearer picture of communication in collaborative design can be gained that tells us how design unity can emerge and evolve on the basis of interaction among participants using heterogeneous systems of representation and action. This understanding is expected to contribute to design studies in general, and to extend the current CSCW conception of shared drawing space to that of shared modelling space in particular.

In contrast to the design studies mentioned in the above, this paper takes a different measurement. It is considered that an investigation of collaborative design can be approached by analysing, not simulated nor controlled but naturally developed, design expressions taken from historical examples of architectural modelling. The intention is that, by examining the common elements and structures from this kind of evidences, the properties of communication and co-ordination in co-operative modelling can then be described in more precise terms. By further enquiring into the relations among the basic conceptual entities, coherent frameworks can explicitly specify some of the computational requirements.

The remainder of the paper is organized as follows. Drawing on design practice in architecture and other engineering disciplines, Section 2 gives an introduction to the notion of design as modelling. Three historical cases as examples of co-operative architectural modelling are analysed in Section 3. Section 4 presents an exposition of communication in collaborative design based on the preceding case studies. The exposition leads to two conceptual frameworks for describing co-operative design modelling, which centre around the interrelations between common images and distributed design developments. Finally, the implications of the current study and a plan for further investigation are briefly discussed in Section 5.

2 *Design as modelling complex objects*

Owing to the large varieties of components as well as the dynamic relations between the parts and the whole, the objects designed by

architects or engineers tend to be complex in nature. However, alternative techniques have been devised to convey and manage the complexities involved in design. As can be observed in many architectural or engineering workshops, designers often use and manipulate physical tokens (e.g. cardboard, strings, paper, wooden blocks and polystyrene, etc.) when they construct models for various purposes. Working with these physical models, designers are better equipped to develop, reflect, and communicate design ideas with themselves and others. Though it can be extremely tedious, model making has been generally considered by design educators and practitioners to be an essential part of the design processes. The construction and use of physical models has been widely observed not only in individual cases[12,13] but also in group processes[7,9].

In addition to making physical models, designers always produce *drawings*. There are intimate relations between the production of drawings and model making. Sometimes models are made after drawings have been produced (i.e. drawings serve as blueprints in model construction); and at other times, drawings are produced on the basis of models constructed (i.e. models serve as referents or primary sources for drawing construction). In perhaps most cases, designers work in quite a mixed manner, that is, designers produce drawings to develop or elaborate design solutions suggested during model construction, and designers construct models to better inform themselves of the consequences of associating or disassociating design ideas explored in drawings. It is therefore reasonable to say that, in designing complex objects, there is a need for constant switching between drawing making and model making. Seen in this context, drawing can be thought as a somewhat abstract form of design modelling.

As has already been pointed out by Tjalve and others[14], a drawing is a model if it is made to demonstrate the following attributes:

- A drawing represents *modelled properties* (e.g. structure, form, material, dimension, surface, etc.)
- A drawing has a *receiver* who is the person or persons to whom the drawing communicates information
- A drawing is *coded* in systems of symbols (e.g. coordinates, graphical symbols, types of projection) known to the receiver

Seen from the viewpoint of design as modelling complex objects, the activity of drawing can, in general, be said to encompass two interrelated aspects: the representation of conceptual structures and the performance of modelling actions. The former decides to a large extent the range of

11 **Peng, C** 'A survey of CSCW designs in shared drawing space', Working paper, EdCAAD, University of Edinburgh June 1992 (in submission)
12 **Janke, R** *Architectural Models*, Academy Editions, London (1978)
13 **Goldschmidt, G** 'Interpretation: its role in architectural design' *Design Studies* Vol 9 No 4 (1988) 235–245
14 **Tjalve, E, Andereasen, M M and Schmidt, F** *Engineering Graphic Modelling: A Practical Guide to Drawing and Design* Newnes-Butterworth, London (1979)

basic construction elements and their properties for design use; the latter, when performed by individuals, can lead to specific design descriptions (depictions).

The above conception of design as modelling is even more meaningful when considering the development of theory and practice in computer-aided design (CAD). The kinds of conceptual structures, as largely embedded in the processes of making physical models or drawings, can be made overt when the models are constructed in an electronic design studio. Modern CAD systems have provided designers with various computational devices to do so. One of the main benefits of doing so is that a dynamic integration of model construction and design production can be achieved. A good example for illustrating this important notion can be found in the MOLE project[15–17], where a general computational 'kinds-slots-fillers' mechanism is provided for designers to represent and evolve their own conceptual structures which can then be used by themselves to instantiate specific design instances. The problem of how to support collaborative design has not been approached by the researchers in CSCW from the modelling perspective described in the above.

Some researchers advocate the importance of *activities* that are observed in collaborative design sessions[18,19]. To meet the requirements for direct communication among collaborators, the supply of real-time supports for group interaction in design (i.e. via talking, gesturing, sketching and writing etc.) among geographically distributed participants is of primary concern. On the other hand, some researchers consider that the structures and organizational uses of *artifacts* play an essential role in mediating collaborative work[20,21]. The design and use of collaboration-supporting tools envisaged in the latter view places great emphasis on the inclusion of formalized structure or knowledge of product that is supposedly agreed by most practicing designers. Given those CSCW findings and experiences, this paper proposes that *design as modelling* can be an alternative perspective for investigating the communicative aspects of collaborative design. The objective of the investigation is clear: to identify the conceptual frameworks that unify the aspect of activities and that of artifacts, such that the basic criteria for defining CAD tools with CSCW features can be articulated.

3 *Examples of co-operative architectural modelling*

Architectural modelling can take place in various dimensions, allowing for a wide variety of collaborative involvements. What follows is an introduction to this variety illustrated by three case studies of co-operative architectural modelling. The first case shows a one-dimensional converg-

15 **Krishnamurti, R** 'A model for design description', *Edinburgh Architecture Research*, Vol 12 (1985) 71–89

16 **Bijl, A** 'Strategies for CAD', in **P J W ten Hagen and T Tomiyama** (Eds), *Intelligent CAD Systems I: Theoretical and Methodological Aspects* Springer-Verlag, Berlin (1987).

17 **White, R** 'FireMOLE – an architecture for reasoning with drawings', *Design Studies* Vol 13 No 3 (1992) 320–334

18 **Bly, S L** 'A use of drawing surfaces in different collaborative settings', In **I Greif** (Ed) *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88)* ACM Press, 1988, pp 250–256

19 **Tang, J C** *Listing, drawing, and gesturing in design: a study of the use of shared workspace by design teams.* Technical Report SSL-89-3, Palo Alto Research Centre, 1989

20 **Lakin, F** 'Visual languages for cooperation: a performing medium approach to systems for cooperative work', in **J Galegher, R E Kraut and C Egido** (Eds) *Intellectual Teamwork: Social and technological Foundations of Cooperative Work*, Lawrence Erlbaum Associates, (1990) pp453–488

21 **Fischer, G, Grudin, J, Lemke, A, McCall, R, Ostwald, J, Reeves, B and Shipman, F** 'Supporting indirect collaborative design with integrated knowledge-based design environments', *Human-Computer Interaction*, 1992, (To appear in Special Issue on Computer Supported Cooperative Work).

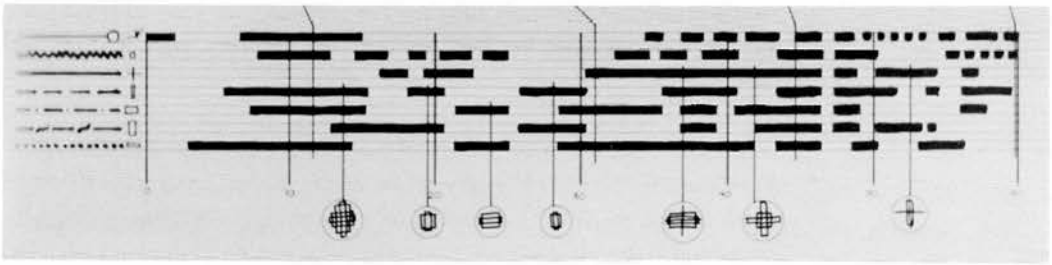


Figure 1 Landscape designers' constructing and using Score *in modelling fountain pattern and actions over a period of time. (Taken from Halprin[22], p.56)*

ence of two individual design worlds employing different conceptual schemes in modelling a fountain design. The second shows the overlaying of two-dimensional diagrams constructed by at least three engineering disciplines, for re-engineering a large industrial building. The third case illustrates a three-dimensional funicular model that was devised commonly, but used differently by individuals who participated in a church design project.

3.1 *Case 1: between score and diagram*

Design project: Seattle Center Fountain, Seattle, USA, 1962–1964.
Design aspects and participants: waterscape design by two landscape architects (Lawrence Halprin, Curtis Schreier); fountain engineering by a mechanical engineer (Daniel Yanow)[22].

The scoring and diagraming spaces:

- The Landscape Architects (LA) used a particular representation scheme called 'score' for modelling fountain patterns and actions in a temporal frame (Figure 1). A score has two dimensions: one for regulating multiple temporal sequences, represented in certain lengths of bars; the other for configuring spatial structures of different fountain stages (platforms), represented as point, square cross, rectangle etc. By manipulating the bars, a score reveals different compositions of active fountain stages against the inactive ones over a period of time.
- The Mechanical Engineer (ME) used 'diagrams' to model mechanical components of piping, jetting, sprinkling design (Figure 2). A pool piping grid was composed in a system of graphical symbols, corresponding to a set of design objects whose attributes were specified in words and numerals. In relation to the piping grid, a mechanical section was constructed to convey sectional information. Due to the correspondence between the mechanical components and the graphic-

22 Halprin, L *The RSVP Cycles: Creative Processes in the Human Environment*, George Braziller Inc., New York (1969) pp 54–57

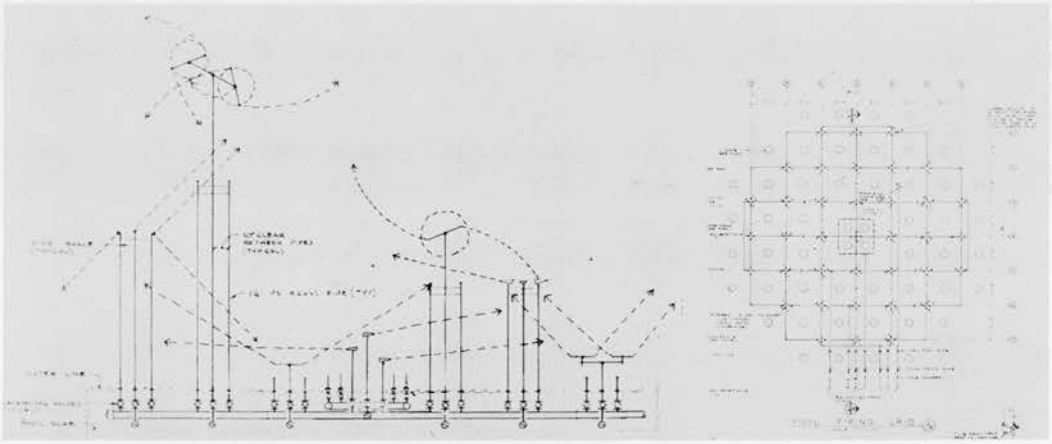


Figure 2 Mechanical engineer's model of behaviours of some mechanical components of fountain design, including piping diagrams and mechanical sections. (Taken from Halprin[22], p.56)

al symbols, the ME could virtually change the attributes and relations of particular design objects by manipulating parts of the diagrams.

A common space for projecting water effects: Figure 3 shows a series of graphical expressions of *squiggles* spreading over a regular grid. This evidence implies that a common modelling space shared by LA and ME was being used, combining the designs in the score and in the diagram, by which a sequence of fountain effects can be *projected*. More importantly, the projected images of the fountain effects can be interpreted both in the LA's view – the actions of fountain stages as scored over a time span, and in the ME's view – the fountain kinematics concerning the motions in pipes, jet heads, and sprinklers, as configured in the piping grid and the mechanical section. Clearly, this is a case of collaborative modelling where a set of *common images* was generated by a certain convergence of

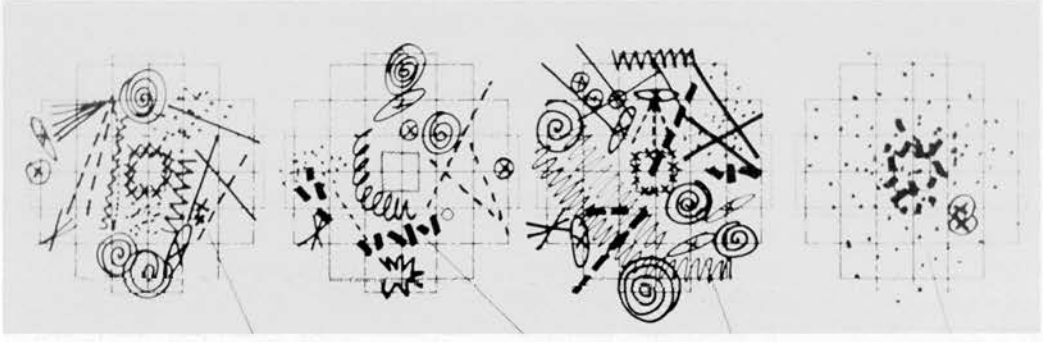


Figure 3 Model of fountain formed by combining LA's scoring and ME's diagraming, it projects water effects, allowing for different interpretations. (Taken from Halprin[22], p.56)

two heterogeneous design worlds, and, once formed, it allows for *interpretations* of the design consequences from different participating viewpoints.

Interrelations between the score and the diagram: Given the above evidence, two interrelations between scoring, diagraming, and projecting spaces are worth noting, which yield further explanations of what constitutes participation in developing the fountain design.

- Sequences of water effects at particular moments cannot be projected solely in the LA's scoring space nor in the ME's diagraming space; the possibility of projecting these effects is influenced by knowing what fountain stages are active then and which mechanical devices are operating on those active stages, plus how they will behave: a convergence between two individual modelling spaces whenever a projection is undertaken.
- Modelling actions taken in individual spaces change not only the state of the score or the diagram but also the state of the common image when projected; ME may take further actions on the changing water effects propagated from LA's actions in changing the score, and vice versa. Communication and co-ordination are called for to resolve any disagreements or conflicts that arise.

3.2 *Case 2: Co-operation through overlay diagraming*

Design project: Cummins Research and Engineering Center, Indiana, USA, 1964–1968.

Design aspects and the participants: Structural Engineering (SE) (The Engineers Collaborative); Lighting Engineering (LE) (William Lam Associates); Mechanical Engineering (ME) (Cosentini Associates). The main design issue is centred on how to 'rearrange ductwork to the structure and to baffle the indirect light sources'[23].

Distributed diagraming spaces: Each engineering discipline had its own object-based diagraming space, employing a particular set of graphical symbols and operations to model building components. There were at least three diagraming spaces participating in the re-engineering project: SE, ME, and LE (Figure 4). Each diagraming space employed a special coding system to represent the modelled building components.

Evolving the environmental design through overlaying: According to Lam[23], the group processes evolved a 'fishbone layout' which proved to be economical and satisfactory to all participants (see Figure 5)[23]. Clearly,

23 **Lam, W M C** *Perception and Lighting as Formgivers for Architecture* McGraw-Hill (1977) pp 125–129

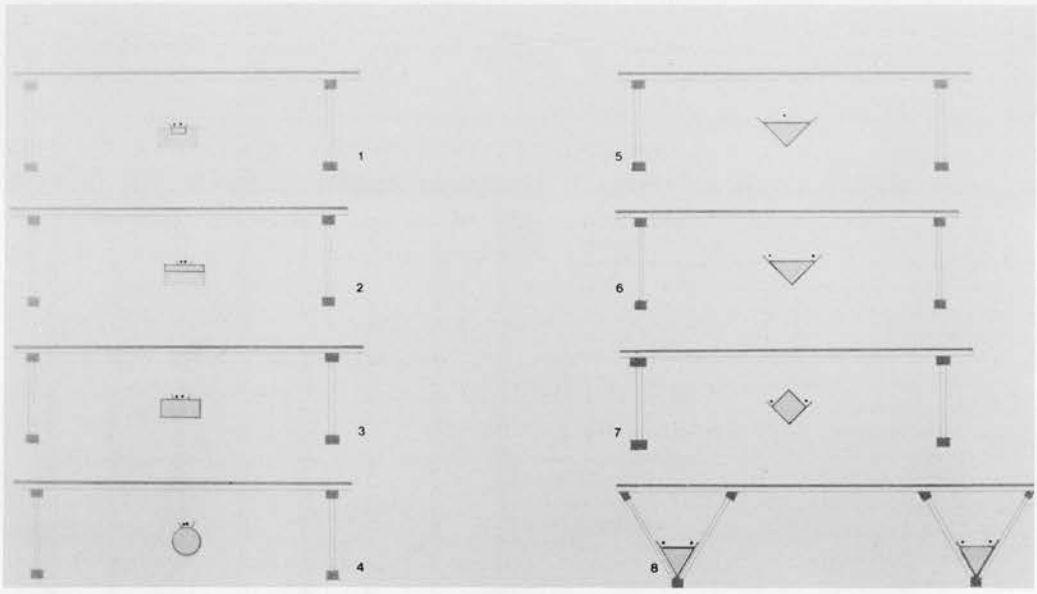


Figure 4 Multiple diagraming spaces in different layers showing participants' heterogeneous coding systems for modelling aspects of building design. (Taken from Lam[23], p.126)

the emergence of the fishbone image is conditioned by the participants' continuously *overlaying* their individual diagrams.

Articulation of common images: Apart from the interrelations observed in the previous case (Case 1), the current case shows that participants can further differentiate a common image into various parts that have different roles and local functions within the emerging whole (the spinal cord and ribs of a fishbone, in this example). Moreover, differentiated portions of a common image can be distributed to individual workplaces, on which the developments of elaborated domain solutions are based.

The current case shows that participation in design is maintained by a *to and fro* relation amongst the individual and the commonly shared. More specifically, by means of overlay diagraming, the overall teamwork process is composed of the following subprocesses.

- *Overlay diagram construction*: a participant can construct diagrams on top of extracted common images which may contain parts of diagrams drawn by other designers working on different aspects.
- *Overlay design checking*: collaborative design can be evaluated by checking overlaid consequences in respect of certain criteria such as the detection of spatial clashes.

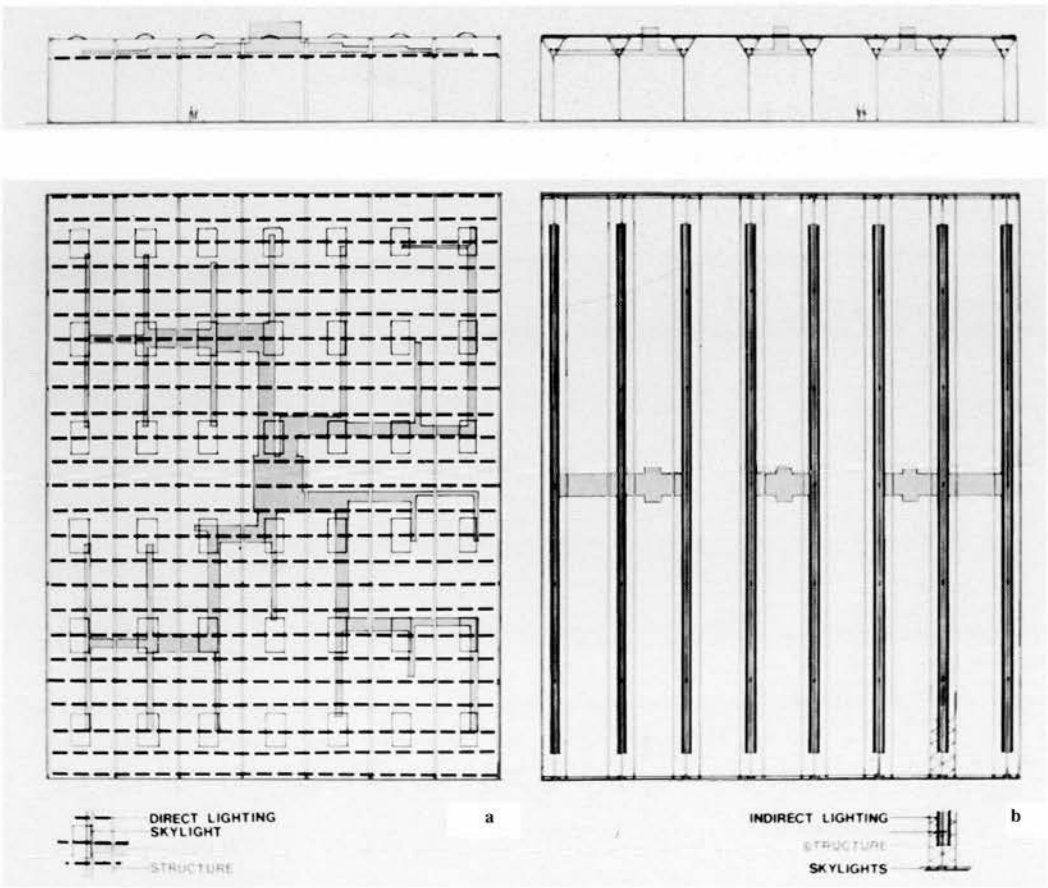


Figure 5 Combined images of structural, mechanical, lighting design solutions evolved through participants' overlaying design developments explored in each domain. (a), earlier design, (b), descendant of (a), with 'fishbone' image. (Taken from Lam[23], p.126)

- *Overlay design amendment*: a participant can modify parts of his or her own diagrams by referring to parts of the diagrams underlaid for various purposes (e.g. geometrical, structural, or acsthetical etc.); and one designer's amendments may cause related changes to be made by others.

3.3 *Case 3: Funicular modelling revisited*

Design project: The Colonia Güell Church, Barcelona, Spain, 1889–1914.
Design aspects and the participants: site planning and structural form by architects (Antonio Gaudí, Jose Canaleta); structural engineering by a civil engineer (Eduardo Goetz); ornamentation by a sculptor (Juan Bertran)[25].

24 **Collins, G R and Nonell, J B** *The Designs and Drawings of Antonio Gaudi*, Princeton, N.J. Princeton University Press (1983) pp 31–35
25 **Martinell, C** *Gaudi his Life, his Theories, his Work* Editorial Blume, Barcelona (1979) p 335

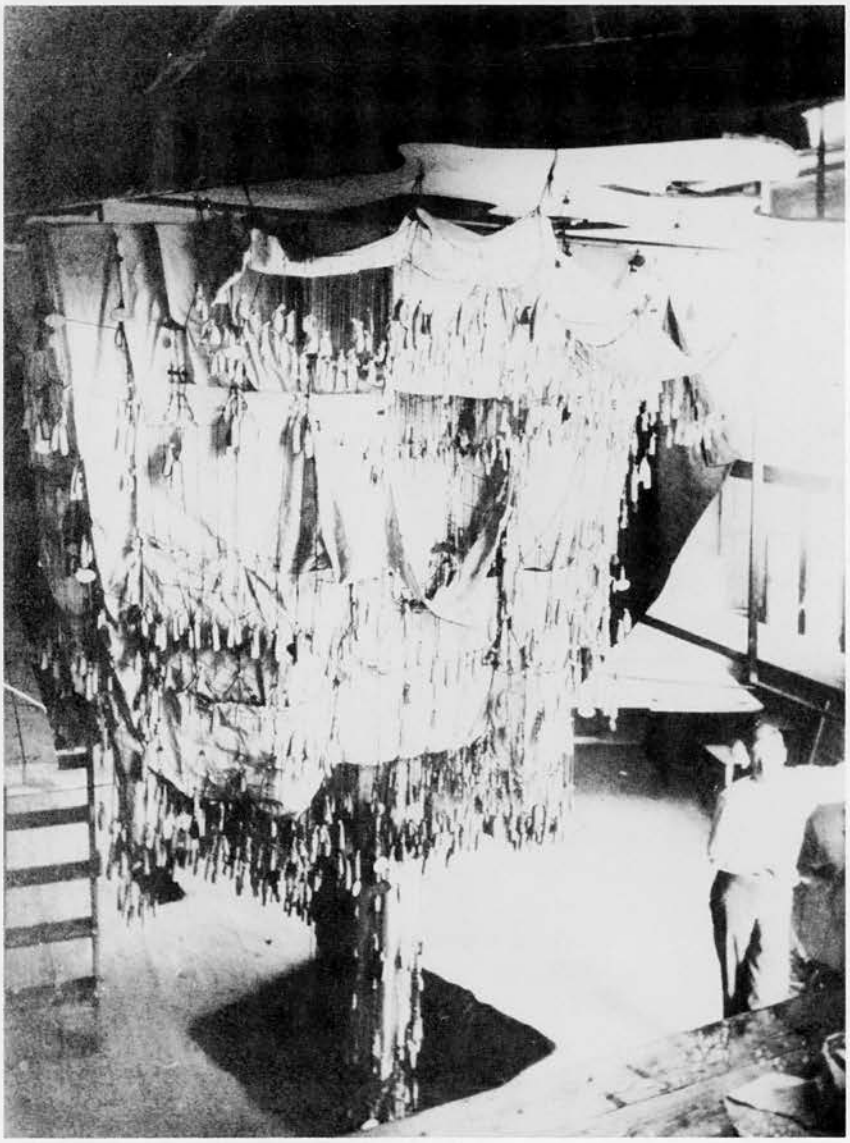


Figure 6 Funicular model constructed from Colonia Güell church project, hanging in workshed. (Taken from Collins and Nonell[24], Figure 39)

The funicular modelling space: An upside-down funicular model was constructed by the design participants at the inception of the project. According to Collins and Nonell[24], this large three-dimensional model, which was shared and manipulated by all participants for different design tasks, had the following distinctive types of model components (Figure 6):

- *Cords* hung in loops corresponding upside down to the placement and shapes of the piers and arches of the building's vault
- Several pieces of irregularly shaped wooden *boards* fixed onto the structure of the workshop, representing contour lines of the building site
- *Weights* made of pellets contained in small sacks (measured in the scale of 1/10 000), when attached to the hung cords, distorting the cords' catenary curves into funicular polygons
- *fabric* (tissue paper) draped onto the web of funicular polygons, representing the volumetric effects of the building exterior
- a set of domain-independent objects consisting of *jointers*, *hooks* and *clippers*, which do not correspond to any particular building components of the church design, but function significantly in connecting the above types of model components and in facilitating changes in parts of the funicular model (Note 4)

From the funicular model to other distributed modelling spaces: Apart from the funicular model constructed by the group, there are other special modelling spaces created and used by different individuals. As reported in the research literature, the graphical evidence shows that the distribution of special modelling spaces includes at least the following.

- The civil engineer's structural calculations: the distribution of loads in space and the thrusts of force lines were calculated by the engineer in a two-dimensional vector space; the funicular model was seen as a three-dimensional illustration of planar and sectional *graphic static calculations*.
- The architects' sketches of the exterior and interior spaces: photographs of the exterior and interior of the funicular model were taken and turned right-side up by the architects as the underlay information for modelling the locations, proportions, and shapes of openings (i.e. the fenestration of the building).
- The sculptor's sketches of the ornamentations: the sculptor was concerned with the design of sculptural objects as the ornaments for the building's exterior and interior; like the architects, he took photographs of the funicular models for his own design purposes and tried out design solutions via overlay sketching.

By gathering the different sets of pictorial evidence, Figure 7 shows an overview of the collaborative setting for the Güell church design: firstly, a common workspace is used to construct and change the funicular skeleton; secondly, a number of separate workspaces are created and used by different participants for domain-specific design developments; and

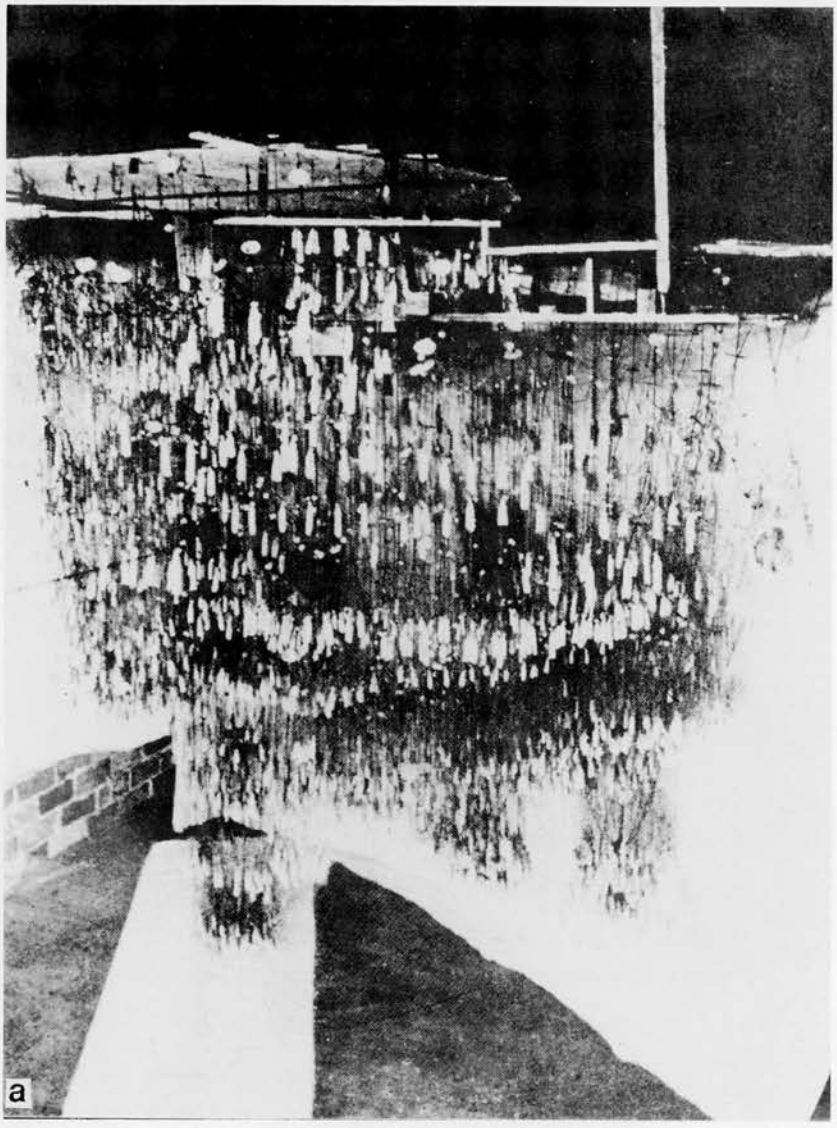


Figure 7 Overview showing number of distributed workspaces involved in Güell church design project (a), funicular model constructed in comma workshop; (b), interior design of church sketched by sculptor on top of inverted photographs taken inside funicular model; (c), exterior design of church sketched by architects on top of inverted photographs taken outside model; (d), force lines constructed by civil engineer on projected elevation for structural calculations. (Taken from Collins and Nonell[24], Figure 41, Plate 57, Plate 55A and Plate 59, respectively)

thirdly, the distributed modelling spaces are related to the funicular modelling space in one way or another.

Group interaction supported by the funicular modelling space: Given the

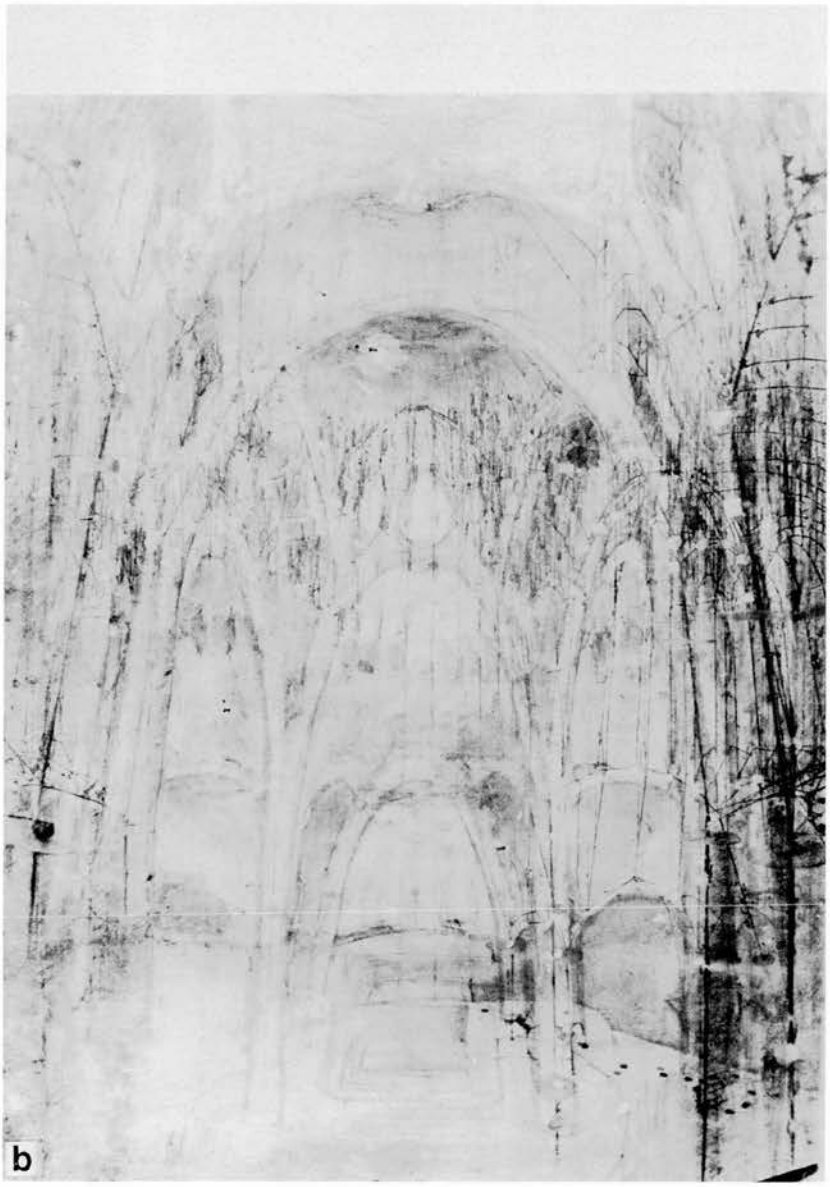


Figure 7b

above observations, several accounts can be given of what makes the funicular modelling space a shared workspace for the design team, and how the shared model serves as evidence of interaction between the participants.

- Firstly, the funicular modelling space was continuously developed and

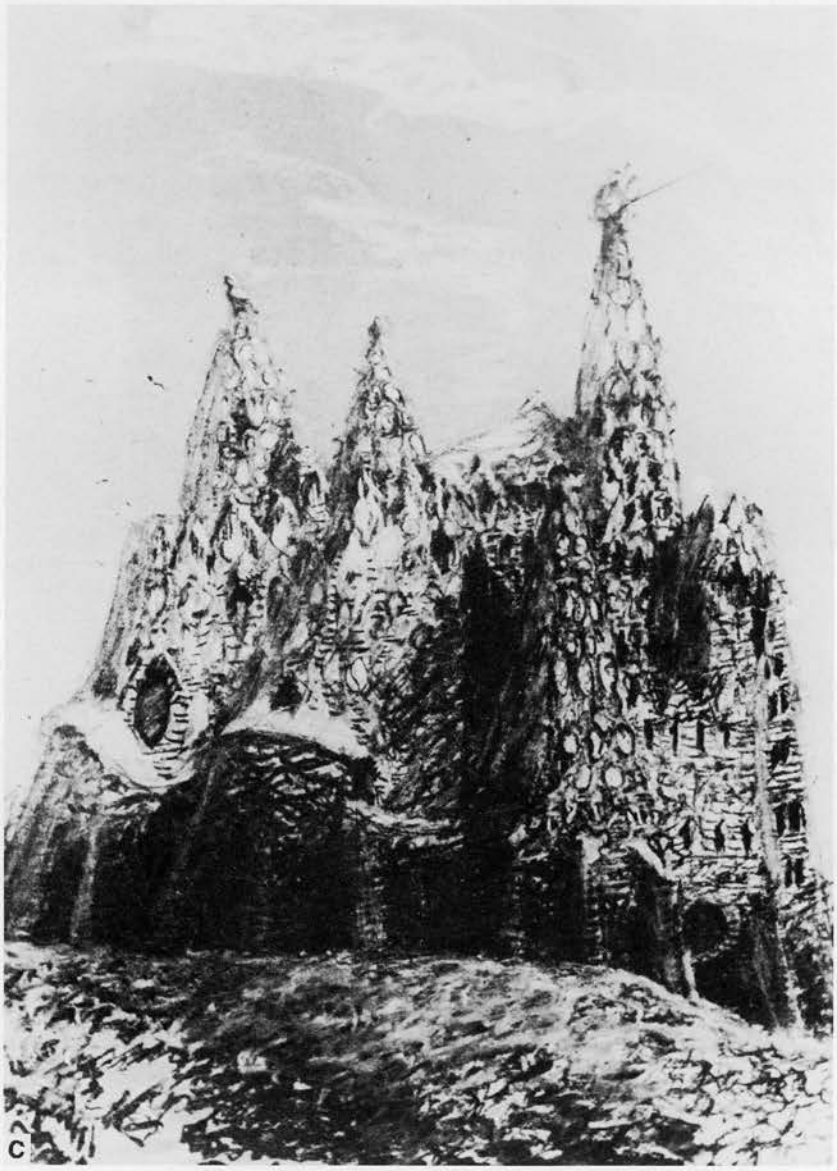


Figure 7c

used by the design team for supporting long-term collaboration; the participants collaborated on delicate exploratory work lasting over 10 years[25].

- Though all participants shared the same construction of a structural form, the shared funicular modelling space allowed them to manipulate parts of the skeleton for reasons other than the strictly structural (Note 5).

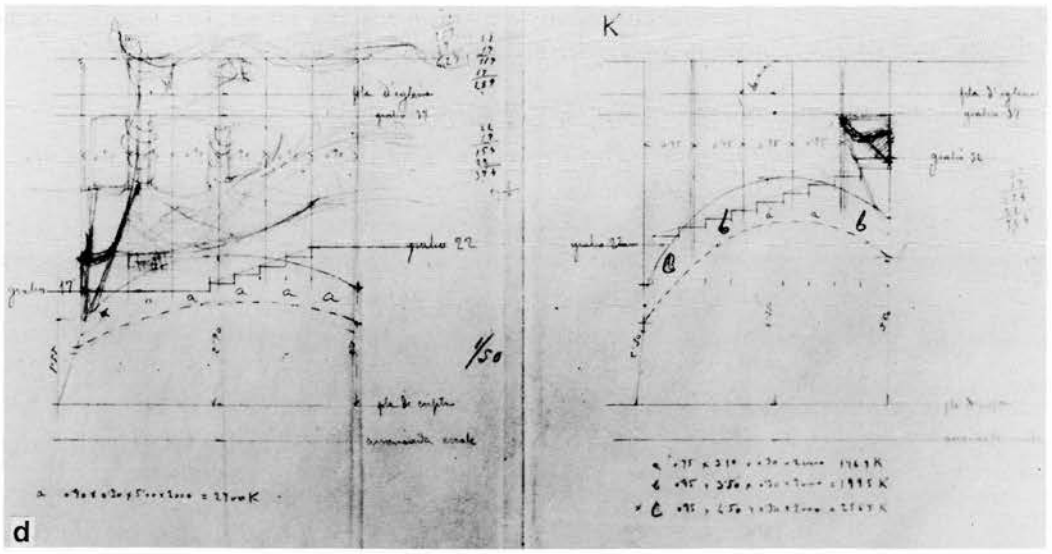


Figure 7d

- For any state of the funicular model, the participants could have individual interpretations and derive design information from, perhaps, different measurements. This information served as the basis for the individuals to elaborate domain-oriented design developments distributed over several workplaces.
- As the earth's gravity was one of the (direct) forces shaping the model, this can explain how the group interaction could be co-ordinated by the shared modelling space. Through the action of G-force, the model constructed and manipulated always conforms to the physical law of *funicular structure*[26]. Therefore, a modelling action taken by an individual, for whatever reason, can stimulate other team members' interpretations and actions in response to the changing state of the funicular model.

4 *Group communication in collaborative design*

Verbal and graphical evidence showing that some forms or processes of communication must have taken place for the participants to arrive at certain integrity in the design results has been presented in the above examples of architectural modelling. The aim of this section is to explore the implications of these case studies, leading to some general notions about communication in collaborative design. However, a presentation of these notions should not be taken as programmatic strategies or models of collaboration applicable to real teamwork practice in design. Instead, a set of basic concepts for exploring some abstract patterns of collaborative

26 **Schodek, D L** *Structures* Prentice-Hall (1980)

settings is our primary concern, which, in turn, may help to articulate the objectives and problems of communication in collaborative design.

4.1 General conditions and goals

Though adopting different modelling approaches to the developments of design solutions, the preceding three examples suggest some general conditions and goals of collaboration. For communication and co-ordination among individual team members to take place naturally, the following general conditions need to be taken into account.

- Specific or concrete goals of collaboration (e.g. definite descriptions or depictions of final design products) are not known to or cannot be clearly defined by any participants at the outset.
- Heterogeneous systems of representation and action employed by individual members are necessarily involved.
- No predefined scheduling schemes can be applied to how participating design disciplines should co-ordinate with each other during collaborative sessions, i.e., the strategies for specifying and satisfying precedence constraints of fulfilling co-operative processes cannot be determined by particular individuals in advance.

Apart from working with the above conditions, designers have a strong obligation or commitment to the production of sorts of design information during or after their joint modelling efforts. Kinds of information produced in the forms of sketches, drawings, physical models, and specifications etc., are the outcomes of collaboration. In particular, two kinds of information may be considered as the general, or metagoals of collaborative design.

- The information conveys shared conceptions of *unity* in design artifacts to be realized via final constructions. The expressions of unity are made on the basis of shared modelling spaces in dealing with general issues, such as form, circulation, human biological and ecological needs, etc.
- The information contains separate records of *specifications* addressing problems which occur in particular design aspects. The design specifications are produced on the basis of separated individual modelling spaces in dealing with technical problems, such as structural, lighting, mechanical services design, etc.

Given the above descriptions, we may ask how designers think and act, under general working conditions, so that the metagoals are fulfilled or emerge from collaboration. A simple answer is that, of course, designers

communicate with each other. However, the perspective of co-operative architectural modelling discussed so far provides a setting for deeper exploration of the properties of communication in collaborative design. An exposition of these properties is given below. The intention is to present a number of elementary concepts that formulate the human and artifactual factors involved in co-operative modelling. By further enquiring into the interrelations among these concepts, the general properties of group communication in collaborative design are described in more specific terms and put into a coherent framework.

As suggested above, design can be characterized as an open-ended process of modelling complex objects. If this proposition is accepted, two related aspects of design follow immediately.

- Design concepts and constructs are continuously introduced and (re)structured as the basic workspaces by individuals or groups – the aspect of forming *modelling spaces*.
- Shapes and nonshape properties of design artifacts are substantiated, manipulated, and evaluated iteratively by individuals or groups – the aspect of performing *modelling acts*.

Therefore, design as an individual or group activity is essentially the performing of modelling acts in modelling spaces. These two aspects which constitute a basic setting for exploring group communication in collaborative design will be discussed more abstractly below.

4.2 *Modelling spaces*

In a longer term collaborative design process involving heterogeneous modelling spaces, as shown in each of the examples, a clear distinction can be made between the modelling spaces that are formed and used by individuals and by the team. Therefore, it can be said that there are multiple individual modelling spaces (IMSs) which are physically and/or functionally separated from a group modelling space (GMS). GMS and IMSs are constructed to hold the creations and modifications of *common images* and *domain design expressions*, respectively.

Common images in a GMS

It has been repeatedly shown in all three cases that participating designers' working with heterogeneous design worlds does not prevent their achieving visual images that can be shared by all the participants (Note 6). Taking various forms, expressions of common images are created and used as either generic structures (e.g. the funicular skeleton) or as specific

instances (e.g. the fountain squiggles), which allows for individual participants to apply different modelling actions.

Constructing common images: It can be seen that, embodied mainly as graphical objects, common images can be constructed in a group modelling space in the following two approaches

- A common image is constructed jointly by all the participants employing a shared representational and operational system: a common image, which serves as a generic conceptual structure, allows each participant to derive and distribute its parts into individual modelling spaces for different modelling purposes.
- Common images are found by the participants combining and integrating domain-specific design expressions with, perhaps, heterogeneous underlying conceptual structures; which serve as the outcome of participation and co-ordination, common images are inspected and evaluated by individuals to reflect on the design consequences from particular viewpoints.

Changing common images: Given an existing common image, its state is subject to unpredictable changes. Depending on how it is formed (in either of the two ways discussed above), changes onto the state of a common image can be effected in two ways. Firstly, changes can be made directly to parts of a common image by any participant, if it is formed in a GMS using a shared representational and operational system; changes made by one individual to parts of a common image may have related consequences as seen in other participants' IMSs. Secondly, changes can be made indirectly to parts of a common image, if it is formed by combining and integrating parts of heterogeneous design expressions produced by participants. That is, the state of a common image held in a GMS is updated whenever one or more participants modify parts of their domain expressions. Thus, changes to a common image caused by one individual in his or her IMS may stimulate further design changes by other participants.

Domain design expressions in IMSs

The design of complex things like buildings can seldom reach a contractible state without continuous development over months or even years. Each of the case studies clearly shows that, in general, a design project can not be accomplished by a single profession. This means that design developments are often carried out in distributed work processes and settings. Design developments lead to the generation of design expressions, and these expressions tend to be domain-specific in the sense that

they may contain information that is only fully comprehensible to people from the same design discipline. Being different from the modelling of common images, domain design expressions are created and stored in a number of logically and/or geographically distributed individual modelling spaces.

Constructing domain design expressions: With regard to the construction of domain design expressions, another abstraction from the case evidence can be made as follows.

1) Domain design expressions are developed by elaborating parts of a common image that have been proposed in a GMS. Participants interpret the state of a common image from different perspectives and trace down *derivative images* for individual purposes. The derived images are then imported into IMSs and serve as the basis for further elaborations, using whatever domain methods the individuals want.
2) Design expressions addressing particular design domains are firstly developed in the participants' IMSs. The initial developments may be independent of each other; but at later stages, some of these expressions are brought by the individuals into a GMS, serving as the basis for joint construction of common images.

Changing domain design expressions: With respect to construction in IMSs, domain design expressions can be modified in two different ways with the following consequences:

1) Where domain design expressions are developed on top of the information derived from a common image, changes in domain design expressions need to be effected and reflected in parts of the common image. The changes thus made may consequently change parts of other derivative images that are distributed in other IMSs.
2) In cases where a common image is formed on the basis of combining and integrating domain design expressions, changes targeted at the latter can be made directly in the participating IMSs. The changes thus made may lead to an updated common image, which, in turn, can motivate further changes to other participants' domain design expressions.

Coupling of modelling spaces

Four concepts in terms of group/individual modelling spaces, common images and domain design expressions have been formulated above. These basic constructs may constitute the artifactual factors of co-operative design modelling. They are described as separate conceptual

entities for purposes of clarity. As has been implied, there are intrinsic relations between these factors. Particularly, in considering the construction and subsequent changes to common images and domain design expressions, necessary exchanges exist between group and individual modelling spaces. For these interrelations, the fifth concept, coupling of modelling spaces, is now formulated more concisely:

- {(*CI*/GMS) → (*DDEs*/IMSs)} That is, the construction and change of common images (*CI*) in a group modelling space leads to (or constrains) the developments of domain design expressions (*DDEs*) in distributed individual modelling spaces. The funicular modelling and the graphical modelling in other aspects shown in Case 3 is an example of such a coupling between group and individual modelling spaces.
- {(*DDEs*/IMSs) → (*CI*/GMS)} That is, the development of domain design expressions in distributed individual modelling spaces leads to (or constrains) the construction and change of common images in a group modelling space. The coupling of the scoring, diagramming and projecting spaces shown in Case 1, and that of the three diagramming spaces together with shared overlapping space shown in Case 2 are two examples of this type of modelling spaces coupling.

4.3 *Modelling acts*

We now explore what *acts* designers perform in the course of modelling design objects. It is widely acknowledged that there are no 'correct' architectural designs. Given the same design task and tools, it is probable that very different outcomes will come from different designers. A reasonable account of the differences may be formed in the element of action. It is through an individual's actions on artifacts that design objects show their essential character. Therefore, as another conceptual deliberation, the execution of modelling actions constitutes *modelling acts*. Some basic design actions, seen from a modelling point of view, are singled out in the following general terms.

- *Representing* – involves first, a collection of *private objects* (constructs) which represents analogically or symbolically the corresponding elements of a design artifact in the real world: secondly, specifying how instances of the primitives are related in terms of what {em operations} are applicable to the constructs. The act of *representing* leads to (explicit) *conceptual structures* in a modelling space.
- *Mapping* – the act of translation and integration of (parts of) an existing conceptual structure to (parts of) another conceptual structure.
- *Constructing* – (given a formalized conceptual structure) the act of

Spaces Coupling / Modelling Acts	*CI* / GMS → *DDEs* / IMSs	*DDEs* / IMSs → *CI* / GMS
Representing	(1)	(2)
Mapping	(3)	(4)
Constructing	(5)	(6)
Querying	(7)	(8)

Figure 8 When situated in the two settings of coupled group/individual modelling spaces, modelling acts become communicative acts in collaborative design (Circled numerals correspond to numbered items in section 4.4)

applying operations to selected primitives for the creation or modification of *CI* or *DDEs*.

- Querying – (given a formalized conceptual structure) the act of applying operations to parts of *DDEs* or *CI* for evaluating design instances.

4.4 *Communicative acts in collaborative design*

Finally, a matrix of *communicative acts* in collaborative design is proposed, which summarizes the properties of group communication identified in this study. The matrix is constructed by putting together the elements drawn from the preceding descriptions of the artifact and the action aspects of collaborative modelling. A key point is that, when situated in the settings of coupled group/individual modelling spaces, modelling acts become design actions that require communications among designers working in different domains (Figure 8). Following the matrix formulated, some of the basic requirements for communication in collaborative design are explained.

1) (Representing in {(*CI*/GMS) → (*DDEs*/IMSs}) The modelling act of representing becomes communicative among participants in the setting up of *shared generic structures* in a group modelling space. This involves the team members' sharing of *group primitives* and the operations for manipulating instances of the primitives such that common images can be evolved.
2) (Representing in {(*DDEs*/IMS) → (*CI*/GMS}) Conceptual structures are set up by the participants in individual modelling spaces; group communications are required when parts of individual conceptual structures are combined and integrated into common sets of constructs in a group modelling space.
3) (Mapping in {(*CI*/GMS) → (*DDEs*/IMSs}) Participants apply deductive or projective means on states of common images and produce

derivative images for individual users. Group communications come into play when a member's mapping parts of a common image *overlaps* (spatially or logically) with another member's mapping.

4) (Mapping in {(*DDEs*/IMSs) → (*CI*/GMS}) Participants translate parts of individual conceptual structures set up in one design domain into another. An individual's knowledge of what parts are to be translated where, relys on communication for resolving interpersonal/group purposes.
5) (Constructing in {(*CI*/GMS) → (*DDEs*/IMSs}) Participants apply operations onto group primitives yielding states of common images in a group modelling space. Owing to the coupling of the spaces, events of creating and/or changing parts of a common image has consequences that may affect members' creating/changing domain design expressions distributed over a number of individual modelling spaces. Group communications are required in case some of the participants refuse to accept the changing states of design expressions that are their domains of concerns.
6) (Constructing in {(*DDEs*/IMSs) → (*CI*/GMS}) An individual applies operations onto his or her own conceptual primitives yielding instances of domain design expressions. Owing to the spaces coupling, the events of creating/changing domain design expressions in individual modelling spaces may lead to changing states of a common image that are observable to all participants. Group communications are called for to resolve disagreements or conflicts thus manifested in the changing states of common images.
7) (Querying in {(*CI*/GMS) → (*DDEs*/IMSs}) In general, querying is to ask for information about the consequences of making or changing states of common images or domain expressions. In this case, designers are concerned with the consequences of changing parts of common images modelled in a group space. The acts of querying become communicative since a true picture of modelling consequences can only be delivered by gathering the states of domain design expressions affected by the events.
8) (Querying in {(*DDEs*/IMSs) → (*CI*/GMS}) In this case, designers are concerned with the consequences of making changes in domain design expressions. Group communications are called for due to a presentation of the current state of common images, which requires participants to provide the latest states of domain design expressions constructed in their individual modelling spaces.

5 *Conclusions*

A spectrum of possibilities: structuralist versus metaphorist. At the beginning of this paper, a problematic situation of collaborative design was

introduced, where designers worked with heterogeneous systems of representation and action. To add more complexities, designers have shared common goals in parallel to those of individual ones to achieve. An interesting issue to explore is how do designers participating in a design project manage to achieve both general design unity and technologically sound domain specifications. For this, we consider that research into co-operative architectural modelling may achieve a better understanding of group communication in collaborative design. From this initial position, the graphical depictions and descriptions from three historical cases were then examined. This evidence has illustrated the heterogeneities and integration–distribution parallelisms in question. By abstracting general constructs from the case material, group communication in collaborative design has been developed in the preceding section.

Two distinct patterns of group communication seem to emerge. To better present this finding, it is useful to characterize the two abstract communication patterns as *structuralist* versus *metaphorist* (Figure 9). Seen from the structuralist stand-point, to collaborate on modelling complex objects, common images as shared generic structures built upon group primitives and operations play a significant role in co-ordinating participants' modelling activities. Group modelling spaces in this case function mainly as a shared construction system, making use of some sort or sorts of physical forces. The funicular modelling space is such an example. However, in serving a similar purpose, there can be other form-finding systems which introduce physical or formal laws other than the gravitational one (Note 7). By mapping parts of common images into derivative images from different design perspectives, domain design expressions can be further developed (or elaborated) in distributed individual modelling spaces. Designers then receive the consequences of making changes by querying states of common images which may activate interpersonal co-ordination.

Seen from the metaphorist stand-point, collaborative design is approached by the participants introducing domain primitives and operations with which domain expressions can be modelled in individual spaces. By presenting domain proposals in a public forum, the collaboration necessary to achieve integrated conceptual structures or schemata is initiated. In this case, common images are collaboratively constructed by combining and integrating domain design expressions, using the shared constructs and operations. Common images are said to serve the participants as shared metaphors, whose states in turn play a role in co-ordinating design activities across various modelling disciplines. The emergence of squiggles and fishbones are examples of common images

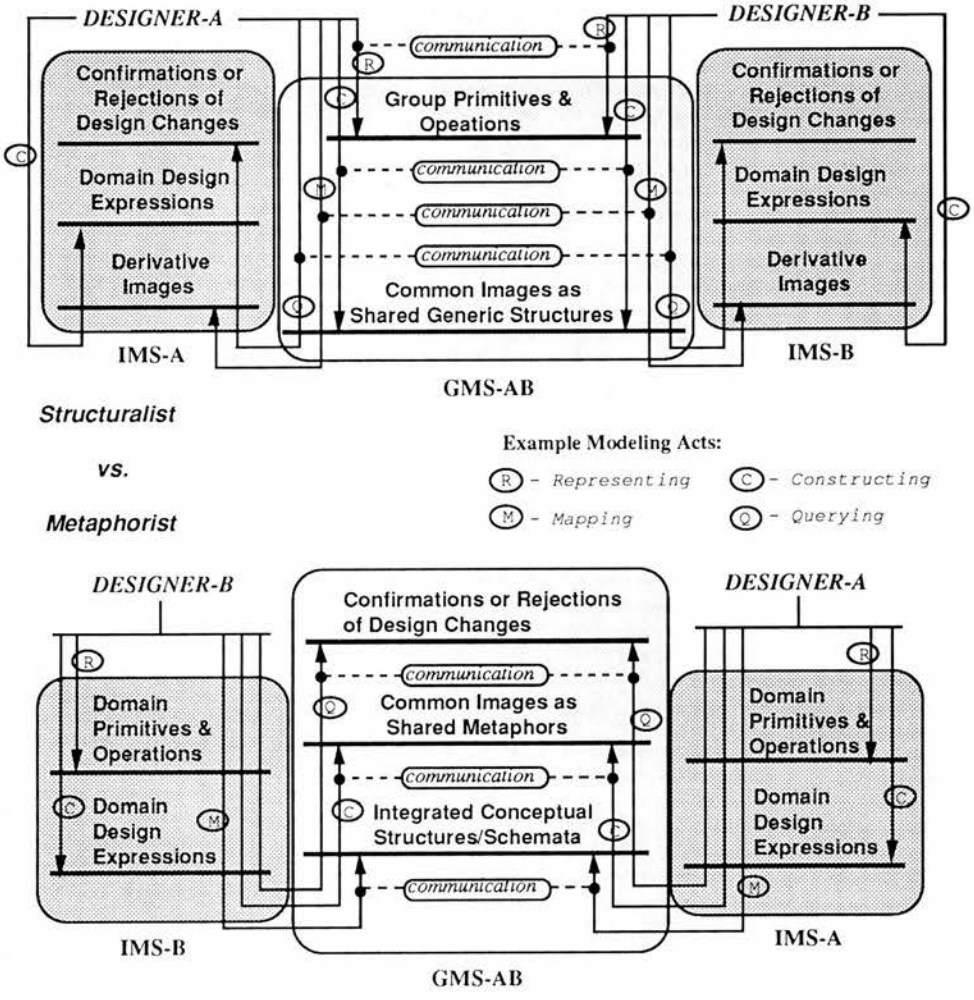


Figure 9 Two abstract communication patterns found in current study of co-operative architectural modelling are characterized as structuralist *versus* metaphorist. *Number of designers indicated is arbitrary. Scaling of 2 to* n *designers in a design team can be envisaged by viewing this diagram as a 'section of a cylindrical structure'. In co-ordinating modelling activities with other members, an individual's workspace is a combination of their own IMS and GMS.*

that reveal the consequences of integrating participating domain expressions in certain ways, as intended by the collaborators.

Further work planned: In several respects, the above abstractions of teamwork patterns in design modelling appear to contrast each other: the platforms and goals for performing modelling acts, the ways and spaces for revealing design consequences, and the processes of generating types of design information, etc. The significance of identifying the two ends of group communication in collaborative design is that it indicates a spec-

trum of possibilities for further research and design on computational supports for collaborative design work

Inspired by the metaphorist approach reported in this paper, a simpler exercise concerning a case of joint shape construction participated by two different design domains was developed and investigated. An initial result has been given elsewhere[27,28]. This formal analysis of collaborative modelling has used the computational framework provided by the approach, an investigation of how common images may be formally represented as shared generic objects based on higher-order parameterized specification is in progress.

6 *Acknowledgments*

The author wishes to thank John Lee and Aart Bijl for their help in developing this paper. This research was partially supported by the UK Overseas Research Students Awards (ORS/9014059).

Notes

1 The term and concept of 'individual object worlds' was introduced and expounded by Bucciarelli and others in studying engineering from an ethnographic point of view[1,2].

2 This observation is mainly drawn from historical studies of design practice and artifacts. For rich and scholarly accounts, see References 3 and 4, among others.

3 For a survey of CSCW-oriented designs in shared drawing spaces with a special interest in reviewing how the issues of supporting collaborative design have been addressed by the research prototypes, see Reference 11.

4 Jointers were used to attaching weights to cords; hooks for connecting the ends of cords to particular locations on the boards; clippers for clipping cords together at various heights (bifurcation).

5 For instance, for the purpose of site planning, cords can be shifted to different hooks or by moving the hooks around the board; for modifying fenestration design, cords can be bifurcated at various heights by sliding the clippers along the force lines; for changing structural form, loads can be redistributed in space by controlling the number of pellets in the sacks or by displacing the sacks' jointers to different positions on the cords.

6 This is clearly shown in the sequence of squiggles drawn in the fountain design project, the fishbone image evolved and reported in the engineering research center project, and the state of the funicular skeleton constructed in the Güell church design project.

7 As far as the construction of spatial structures or skeletons is concerned, there are other eminent formal form-defining systems such as the ancient Greek taxis schemas of subdividing a building composition[29] and the more modern spatial grammars devised by Durand[30].

27 Peng, C 'An algebraic specification approach to joint shape construction', Working paper, EdCAAD, University of Edinburgh, August 1991

28 Peng, C 'A formal perspective on teamwork in design modelling', *Edinburgh Architecture Research*, Vol 19 (1992) 137–154

29 Tzonis, A and Lefaivre, L *Classical Architecture: The Poetics of Order*, MIT Press, Cambridge, MA (1987) pp 9–25

30 Durand, J N L *Precis des lecons d'architecture donnees a l'Ecole royale polytechnique* Unterschneidheim, Ger.: Uhl, 1975. Reprint of the 1819 edn published by the author, Paris, issued with the author's *Partie graphique des cours d'architecture*

Computer Supported Cooperative Work (CSCW) **1**: 197–228, 1993.

Survey of Collaborative Drawing Support Tools
Design Perspectives and Prototypes

CHENGZHI PENG*
EdCAAD, University of Edinburgh, 20 Chambers Street, Edingburgh EH1 1JZ, UK, peng @ uk.ac.ed.caad
**Current Address: Information Technology Research Institute, University of Brighton, Lewes Road, Brighton BN2 4AT, UK. Email: cp45 @ uk.ac.bton.itri*

(Received 30 June 1992; in final form 10 May 1993)

Abstract. Along with recent experiments in the design of communication or computer tools for supporting various kinds of group working, the development of *collaborative drawing systems* has emerged as a notable research area within the field of Computer-Supported Cooperative Work. This paper reports a survey of the experiments in collaborative drawing support tools with an objective of reviewing how the issues of supporting *collaborative design* have been addressed by the research prototypes. The survey is presented in three parts: (1) findings from the observations of group interaction in drawing and design activities, (2) a framework for classifying the design issues experimented with by prototypes developers, and (3) a categorisation of the current prototype systems by interrelating the patterns of group use observed with the system features classified. The survey indicates that there are currently at least three different strategies of developing collaborative drawing support tools, which reflect the existence of diversified understanding and technological responses to what and how human collaboration in design may be supported.

Key words. Collaborative Drawing Tools, Shared Drawing Space Activity, Collaborative Design

1. Introduction

Design as a human activity is pervasive; and designers often work jointly to develop usable and meaningful artefacts. The problem of how to develop communication and computer systems that can support collaborative design or problem solving has become an active research area, attracting researchers working on various perspectives. In a recent bibliographical survey of the research literature appeared in the field of Computer-Supported Cooperative Work (CSCW), Greenberg (Greenberg 1991) has formally introduced the key word *shared workspace*. It is noticeable that a large portion of research work on shared workspaces has to do with building prototypes of *collaborative drawing support tools*. Being perhaps inspired and guided by earlier observational studies of working group graphics and shared drawing space activities, (see Lakin 1983; Bly 1988; Tang and Leifer 1988 among others), researchers have been attempting to design and implement prototypes of shared drawing systems. The requirements for these systems to meet are different from the tradition ones; one of the major goals of implementing shared drawing tools is to facilitate communication and coordination among participants in the course of creating and using technical or non-technical drawings.

However, group interaction in design has been observed, described, and analysed differently due to the various perspectives adopted. There accordingly appears a diversity of understanding as well as assumptions as to what might constitute a shared workspace which enables human communication and coordination in carrying out design tasks. Differences in the basic investigation of the question

how are drawings as shared artefacts, and group drawings as shared drawing space activity, related to collaborative design processes?

have resulted in varied technical approaches to answering

what is to be facilitated by shared drawing support tools?

Though a number of group drawing support tools have been previously reviewed (see, for instance, Lu 1992: 92–41), this survey, by reviewing a wider range of group drawing tools, is intended to be more comprehensive. Instead of giving descriptions specific to particular system implementations, the objectives of our survey are (1) to identify the important aspects of understanding collaborative drawing and design activities, regarding the original studies that have been made, (2) to present an framework for classifying the design issues being experimented with by the current prototype developers, and (3) to catalogue the features and components of the prototype systems in the survey.

The remainder of the paper is organised as follows. A discussion of understanding group drawing activities from various perspectives is given in the next section. Given the conceptual aspects discussed, Section 3 is devoted to a classification of the design issues emerging from the current experimentation in collaborative drawing systems. In Section 4, to give a clear overview of the current status of prototype development, a categorisation which mixes the dimension of the modes of system use with the dimension of system components is presented. Finally, some potential topics for further investigation are discussed in Section 5.

2. Aspects of studying group drawing and design activities

Most developments of collaborative drawing tools were motivated and guided by the current understandings of group drawing and design activities. It is therefore an appropriate starting point to look at what has been said or characterised about these activities. Table 1 is a summarisation of nine such studies. The nine groups' studies are chosen because they present original research perspectives and respond to the problems with various prototype solutions. The differences arisen here are significant is showing that it is quite possible to have very

Table 1. A tabulation of nine studies of group drawing and design activities which have direct influences on their own or other prototypes system developments.

Research groups	Fields of observations	Group size	Research methodology	Important findings	Research prototypes
Lakin et al. (Lakin 1983; Lakin 1988; Lakin 1990)	Informal and formal working group graphics in engineering, geological survey, and computing	Oner or more facilitators with non-specified group size	A linguistic analysis of working groups' text-graphics images and manipulation	Spatial and temporal structures in manipulation of text-graphics	*vmacs*, and *Visual Languages for Cooperation*
Stefik et al. (Stefik et al. 1987a; Tatar et al. 1991)	Small teams of computer scientists engaged in face-to-face meetings	2–6 persons	The setting up of an experimental meeting room by networking PCs and a large screen	The effects of turn taking systems on collaborative work taken place in a meeting room	*CaLab Boardnoter Cognoter Argnoter*
Bly et al. (Bly 1988; Minneman & Bly 1991)	Informal drawing in computer user-interface design problems	2–3 designers	Video-audio protocol and a drawing events/ action/clusters analysis	The uses of drawing surfaces in different collaborative settings	*Commune*
Tang et al. (Tang & Leifer 1988; Tang 1991)	Informal drawing in computer user-interface deisgn problems	3–4 designers	Video-audio protocol and an action-function framework for protocol analysis	The importance of the process of creating and using drawings	*VideoDraw VideoWhite-Board*
Ishii et al. (Ishii 1990; Ishii & Arita 1991; Ishii et al. 1992	Informal drawing and computer generated images in computer systems design	2–3 designers	Intuitive understanding and building various versions of experimental	The importance of integrating social protocols with shared workspaces	*TeamWork-Station ClearFace ClearBoard-1 ClearBoard-2*
Fischer et al. (Fischer et al. 1991; Fischer et al. 1992; Reeves et al. 1992)	Formal coding systems and the structures of expert knowledge in the designs of kitchens and computer networks	not specified	Observing design experts in work, then developing a conceptual framework and a demonstration system	The integration of the design of the artefact (made in combined graphics and (texts) and the communication among designers	*JANUS NETWORK-HYDRA XNETWORK*
Lu et al. (Lu & Mantei 1991; Lu 1992)	Multi-layer informal drawing in architectural layout design	2–4 designers	Generating requirements by mapping design behaviours onto skeleton design	Fifteen user requirements for a drawing system to support idea management	*CaveDraw*

Wolf et al. (Wolf et al. 1991; Wolf et al. 1992)	Informal small group meetings for the conceptual design of a "Clutter Collector Robot"	2–10 participants	Developing a pen-based prototype system, putting it in group uses, and getting detailed user feedback	The potential meeting process gains/losses, and the areas for aiding search and retrieval of information	*We-Met*
Brinck et al. (Brinck & Gomez 1992; Brinck 1993)	The remnants of office whiteboard conversations made for technical versus administrative uses	2–3 participants	Recording and analysing sketches, writing, and retrospective descriptions of conversations	A classification of whiteboard objects and their semantic properties in conversational use	*Coversation-Board*

different angles when characterising what are involved in group drawing activities.[1]

To better understand the various issues raised by these original studies and experimentation, a more detailed comparative study is presented below, focusing on the aspects of *events*, *information*, *tools*, and *ownership*. These four aspects are chosen because they are the common conceptual issues that were addressed by the research groups to various extents. Other issues concerning more of the aspects of system design and implementation are left to the next section.

2.1. EVENTS: COLLOCATED VS. REMOTE; SYNCHRONOUS VS. ASYNCHRONOUS

Since any collaborative drawing or design activity must take place in space and time, the patterns of events can be generally differentiated in terms of four basic spatiotemporal structures[2] (Fig. 1.).

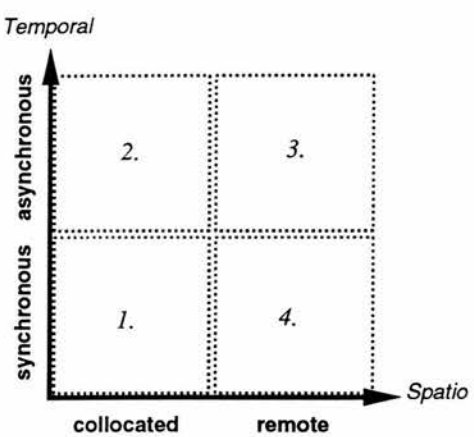


Fig. 1. A spatio-temporal frame for classifying the events of group drawing or design activity into four basic patterns of collaboration. (The numerals correspond to the enumerated items described in the main text.)

1. *collocated synchronous*. All participants taking part in a shared design project work in the same setting via face-to-face simultaneous interactions. A typical group process of this pattern is a design meeting or brainstorming session which demands close physical proximity for intensive communication among group members. As studied by Lakin (Lakin 1983; Lakin 1988), *working group graphics*, operated by one or more operator (or facilitator), plays an important role in aiding collocated simultaneous interactions.
2. *collocated asynchronous*. Given the same projects to embark on, members of a design group are located in the same setting (say, a large design studio) through *indirect* communications over a period of time. Though not being a substantial observational study, a scenario of what might be involved in a day of office work where five staff members take part in a telescope engineering project was described by Lakin (Lakin 1990). What Lakin construed is that participants work in the same setting but are left alone to concentrate in different aspects of the project.[3] In this way, teamwork is carried out mostly via indirect communication among collocated participants (e.g., passing working documents or CAD files to one another not necessarily involving face-to-face meetings).
3. *remote asynchronous*. In carrying out shared design projects, participants work in geographically distributed settings through indirect communications. As a rationale of designing computer tools supporting design teamwork of this pattern, Fischer and others explained that remote asynchronous collaboration can be commonly seen in modern technologically oriented design projects (Fischer et al. 1992):

   Meetings and other types of direct communication are the commonly used means for coordination and collaboration, but in many situations – especially ones involving long-term collaboration – these are not feasible. Modern design projects can extend over many years and can involve a high turnover in personnel. People who are not in the project group at the same time need to coordinate and collaborate in the design of a system.

4. *remote synchronous*. Geographically, team members are separated (from a few feet to, perhaps, thousands of miles away); but they are enabled to have direct communication while making drawings. Events of this type attract most CSCW researchers' attention. In fact, a large proportion of the research prototypes are dedicated to supporting group drawing activities of this type. Under the spatio-temporal circumstances, collaborative drawing activities are made possible by providing participants with shared *virtual drawing spaces* which emulate as much as that can be achieved in a direct face-to-face interaction. In their studies of supporting remote synchronous group interaction in designing or drawing, the following concepts have been forward:

- Drawing activities once shared (simultaneously) can pull designers together and increase their attention and involvement in the design task (Bly 1988);
- The processes of creating and using drawings can convey information that is as important as, or, actually, not found in, the resulting drawings (Bly 1988; Tang 1991);
- Designers are themselves skilled at coordinating communication, and the social protocols acquired via the face-to-face communication can serve the needs of constructive collaboration in the shared workspace (Tang 1989; Ishii and Ohkubo 1990; Ishii and Artia 1991; Ishii and Kobayashi 1992).

2.2. INFORMATION: ACTION-ORIENTED VS. REPRESENTATION-ORIENTED

Collaborative work in design or problem solving in one way or another has to do with the generation and exchange of information. However, different perspectives have varied considerations of what information is to be captured and transmitted. On most occasions of synchronous collaboration, information useful to group interaction is considered to be *action-oriented*; that is, actions recorded or tracked by devices such as video imaging, shadow projecting, mouse movements and so on, are the kind of information that participants may generate, recall, interrupt, and share.

Actions that have been studied in group drawing activities include sketching, writing (listing alpha-numeric text), talking, gesturing, gazing (making eye-contact). The sharing of action-oriented information is claimed to foster and maintain *group awareness*; but how is collective awareness related to the performance of collaborative work? Recently, Ishii and Kobayashi commented on how 'gaze awareness', as supported by the *ClearBoard* system (see Fig. 4), affects two participants' solving the 'missionaries and cannibals' puzzle (Ishii and Kobayashi 1992):

> Through this experiment we confirmed that it is easy for the players to say which side of the river the partner is gazing at and this information was quite useful in advising each other.

In less direct collaboration, on the other hand, synchronous or asynchronous sharing of design ideas or knowledge conveyed by participants in some representation forms is considered more important. When group work involves more technical matters (e.g., the production of technical drawings or the performing of graphical modelling in engineering design), collaboration may necessarily involves some formal systems for constructing and interpreting individual or collective expressions. The views in favour of representation-oriented collaboration have presented the following points, arguing how formal representations of information (graphical as well as non-graphical) may serve group interaction in design:

- In Lakin's view (Lakin 1986; Lakin 1990), spatial and temporal structures (schemata) can be observed in parts of designers' creating and manipulating text-graphic expressions. These structures, on the one hand, limit the kind of expressions and arrangements one may make, while, on the other hand, provide the basis for well-defined spatio-temporal regularities available for automated machine interpretations. Various *visual languages* for collaborative working such as co-authoring, brain-storming, and task structuring can therefore be formally defined.
- In the view of Fischer and others (Fischer et al. 1992), long-term indirect collaborative design takes the model that coordination of individual work in groups is achieved by the individuals' interactions with 'group memory', which can be represented formally as

" a collection of shared information repositories containing a cumulative record of rationale, solution components, information about prior projects, and other information resources for collaboration."

Given the group memory represented both in (hyper-) textual and graphical forms, indirect communication among designers can be supported by the computer-based methods of 'argumentation'[4] and 'critiquing'[5].

2.3. TOOLS: HOMOGENEOUS VS. HETEROGENEOUS

The third aspect of understanding collaborative drawing activities is concerned with the use of tools. The problem lies in whether all participants work with the same set of tools, or each of them may need to operate different sets of tools. There appears again a dichotomy between *homogeneous* an *heterogenous* sets of tools used by participants. Heterogeneity of tools may be due to, for instance, being manual or computer-based, the structures of drawings produced, ways of storing and retrieving data, actions involved in manipulating expressions, or domains of interpretation and so on. Referring to Tang's, Bly's, and Fischer's studies (see Table 1), we may first point out two main arguments why the provision and use of homogeneous sets of tools are considered as being sufficient for supporting collaborative drawing and design:

Synchronous group interaction does not involve domain-specific information or knowledge; i.e., participants converse with each other on design issues readily supported by sufficient common sense such that 'pencil and paper' types tools can satisfy the communication needs of the group.

Participants come from more or less the same professional background and work within the same design domain; i.e., there is no need for streams of

expert knowledge across different design disciplines in the course of collaboration. It is therefore sufficient for all participants to operate the same set of tools, even if it is a highly sophisticated one.

Taking a rather different view, Ishii and Miyake introduce the concept of 'open shared workspace', expressing that "group members should be able to use a variety of heterogeneous sets of tools (computer-based and manual tools) in the shared workspace *simultaneously*" (Ishii and Miyake 1991). And according to Lakin's observation (Lakin 1990), in ordinary office work, there exist *group-individual mode switching*, (i.e., participants sometimes work as individuals and sometimes as members of a group), and *technical task switching* (i.e., work change between various tasks requiring special technical support). The design of various visual languages for cooperation is aimed to provide heterogeneous analytical tools, to which not all participants need to pay equal attention. Lakin also believed that the availability of a general-purpose text-graphic editor, together with multiple special-purpose analysis tools will enable the team to switch between discussing general issues and dealing with more technical details during a meeting session.

2.4. OWNERSHIP: GROUP VS. INDIVIDUAL

There can be no groups without individuals, and this is largely true even if group members have the same background and work with common languages and tools. As a need or an obligation, an individual's identity is basic to the concept of ownership in a context of group work. A drawing created by a group member may not necessarily be *owned* by the individual, if other members are allowed to change or remove it all will. Shown by the observational studies, some researchers suggest that participants themselves are good at co-ordinating individual activities so that there is no need to provide extra facilities for controlling ownership; some others implicitly or explicitly address the issue of supporting the preservation of ownership control to prevent potential malicious or accidental acts such as removing individual/group work results.

Borrowing from the model of 'permissions' found in many multi-user operating systems, the levels or degrees of ownership can be defined in terms of two dimensions: *identities*, and *operations*. For identities, these can be further divided into, for example, personal, sub-group, group, all; in operations, there can be a differentiation between read, write, and execute. This file-based permission model however, can only partially illustrate the ownership issue in shared drawing space; a more fine-grained framework is needed.

As reported in Lu's (Lu 1992) study of teamwork in architectural design, three user requirements were identified, revealing the need for 'seamless and dynamic' transitions between group and individual ownership:

" • Allow participants to declare any portion of a sketch as private and not subject to deletion by others.
• Allow participants to identify, with no additional interaction, who owns a specific design sketch.
• Allow participants to bring in their own ideas from a private drawing surface provided by the shared tools or from a private file to a shared drawing surface. "

Given the different consideration of how ownership may be defined and managed, there are currently three kinds of approaches: (1) the building of multi-user interface components, (2) the design of arbitration algorithms, and (3) ownership embedded in separate drawing spaces. To give some examples, the Cave Draw drawing surface, developed by Lu and others (Lu and Mantei 1991; Lu 1992), has the distinction between 'pencil' input (for producing pencil marks that can be changed by any participants), and 'marker' input (for producing marker marks is distinctive colours owned by individuals); an operation of 'cut-and-paste' is further provided to enable transitions of design ownership during collaboration[6].

In arbitrating the potential conflict of multiple users' grabbing the same *drawing object* simultaneously, the design of GroupDraw (Greenberg and Bohnet 1991) regulates ownership into various levels (see Section 3.1. for a more detailed discussion). Being rather as a technological consequence of video-based or fused computer-video shared drawing spaces, (e.g., VideoDraw, TeamWork Station), a participant naturally owns what he or she draws on an individual screen or desktop surface, since no one can change or erase other participants' work simply by viewing or pointing at them no one's own drawing surface. Ownership, in this approach, is inherent.

To summarise the above discussions, Table 2 gives an overview of the aspects of understanding shared drawing space activities. It is shown that some research aspects are comparatively less explored either empirically or conceptually than others.

3. Prototypes developments and system features

In the above, we have given a review of the current researches in the aspects of group drawing or design activities. Motivated or guided by the various understandings of *what*, CSCW researchers have worked on the problem of *how*, i.e., the development of prototype systems and the demonstration of using these tools in various contexts of group working. In this section, a survey of the system issues arising in the current prototype developments is presented. As a result of this survey, five clusters of system design issues are classified: (1) structures of graphics, (2) network configurations, (3) information storage and retrieval, (4)

Table 2. An overview of the aspects of understanding shared drawing space activity investigated by the different research groups in the survey.

Research Aspects		Lakin et al.	Stefik et al.	Tang et al.	Bly et al.	Ishii et al.	Ficsher et al.	Lu et al.	Wolf et. al.	Brinck et al.
Patterns of Events	collocated & synchronous	◐	●	○	◐	●	○	○	●	◐
	collocated & asynchronous	◐	○	○	○	○	◐	○	○	○
	remote & synchronous	○	○	●	●	●	○	●	○	●
	remote & asynchronous	●	○	○	○	○	●	○	○	○
Information	action-oriented	◐	○	●	●	◐	○	●	●	◐
	representation-oriented	◐	●	○	○	◐	●	○	○	●
Tools	homogeneous	○	○	●	●	○	●	●	●	●
	heterogeneous	●	●	○	○	●	○	○	○	○
Ownership	group	◐	●	○	●	○	●	●	●	●
	individual	●	◐	●	○	●	○	●	○	○

LEGEND: ● Addressed ○ Not Addressed ◐ Addressed to an Extent

multi-user interfaces, and (5) other dialogue channels. The classification, which emerged from our comparative study of the prototype systems reported, is not intended to be exhaustive. Nevertheless, to our view, it is in dealing with these issues that components of experimental collaborative drawing systems were introduced and put together. A discussion of these issues is given in the subsections below.

3.1. GRAPHICS PRIMITIVES AND OPERATIONS

Drawings as visual objects, created and passed around among people, are often constructed from some graphics primitives which may or may not have computational representations within a drawing system. In computer-based drawing surfaces, users are provided with drawing functions for making marks or constructing graphics objects such as lines, rectangles, circles etc. The provision of drawing primitives and functions determines the properties of a drawing surface to a great extent, since these primitives delimit what pictorial expressions

are allowed, and what drawing operations can be applied to parts of these pictures. Certainly, the design of shared drawing spaces is not an exception to this basic principle; but the requirements for supporting possibly concurrent multi-party interaction in shared drawing space have motivated different views on what drawing primitives and operations should be provided. Five different *structures of group graphics* are found on this basic issue.

1. *video-captured images of freehand sketches.* By using markers directly on drawing surfaces, drawings are simply participants' freehand sketches. To transmit the images of sketches made by team members located in different workspaces, video monitors, video cameras, projectors, and networks are set up as working suites. Since there are no computational representations of drawings involved, participants can use white board markers to draw freely whatever they want. Supported by video networks, what appears on an individual's drawing surface is a synthesised *visual space* containing a translucent overlay of his or her sketches and video-captured images of those by others. Apart from physically erasing and taking pictures of the marks left on the drawing surfaces, little can be done on the drawing once made. Figs. 2 to 4 show three design examples of (purely) video-networked drawing surfaces.
2. *pixel-based graphics.* Pixel-based (or bit-mapped) graphics is often defined as the picture representation of drawings as arrays of pixels on computer

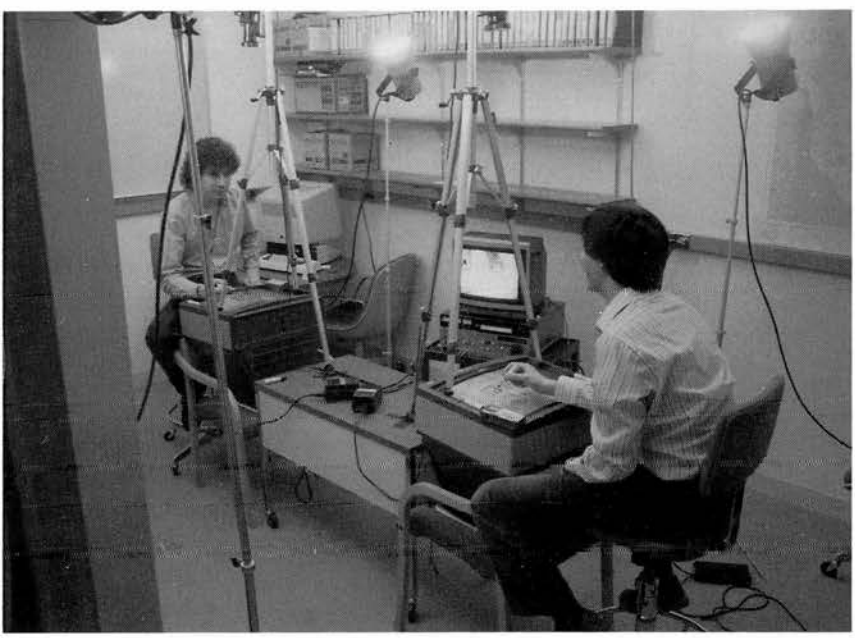


Fig. 2. The drawing surface of VideoDraw, developed at Xerox PARC, used horizontal 20' video monitor screens with dry-erase ink markers [Source: Fig. 2 of (Tang and Minneman 1991a)].

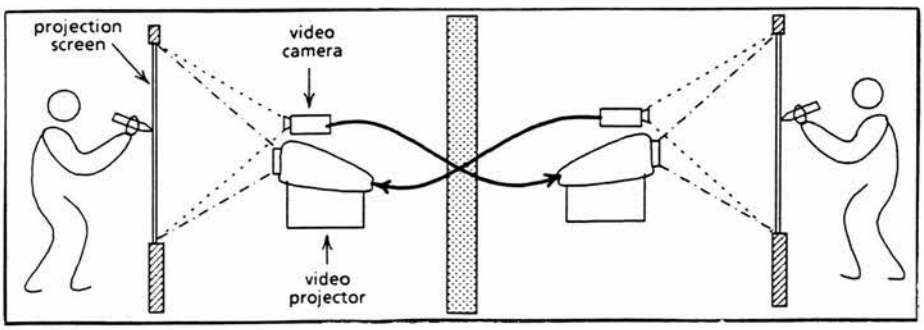


Fig. 3. The shared drawing space of VideoWhiteBoard, also developed at Xerox PARC, used wall-mounted rear projection screens (approximately 4.5' by 6' with standard dry-erase whiteboard markers. [Source: Fig. 5 of (Tang and Minneman 1991b)]

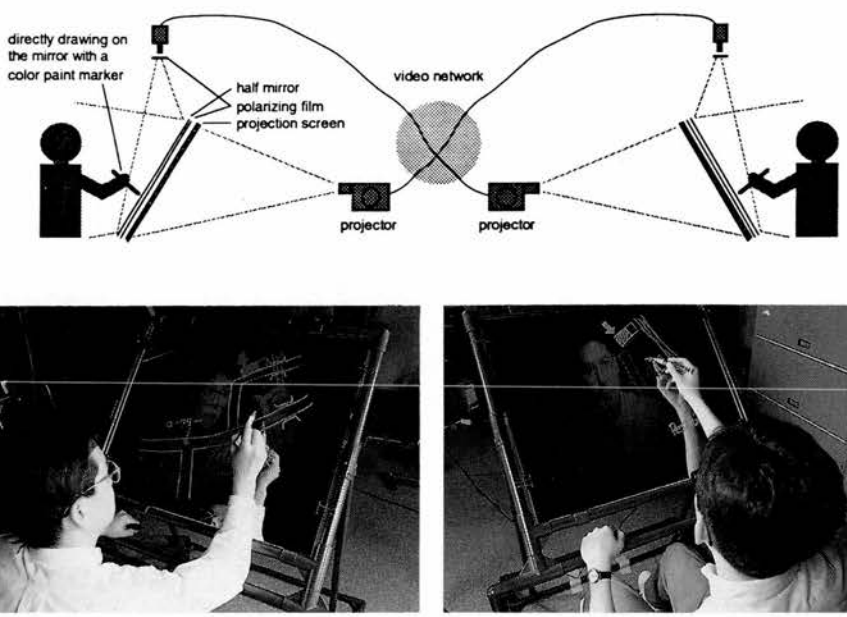


Fig. 4. The shared drawing board of ClearBoard-1 developed at NTT was composed of a projection screen, a polarising film and a half-silvered mirror with water-based fluorescent paint markers[7]. [Source: Figs 4 & 5 of (Ishii, Kobayashi and Grudin 1992)]

screens, which corresponds closely to the data storage pattern in computer memory. As the primitive of most *painting* systems, a pixel has only the states *on* or *off*, and it has no relation to the states of others. Therefore, drawings in pixel-based graphics have typically no underlying structures or models specified by the graphics system. Drawing functions can be implemented as procedures for generating images of arbitrary marks, lines, rectangles, circles

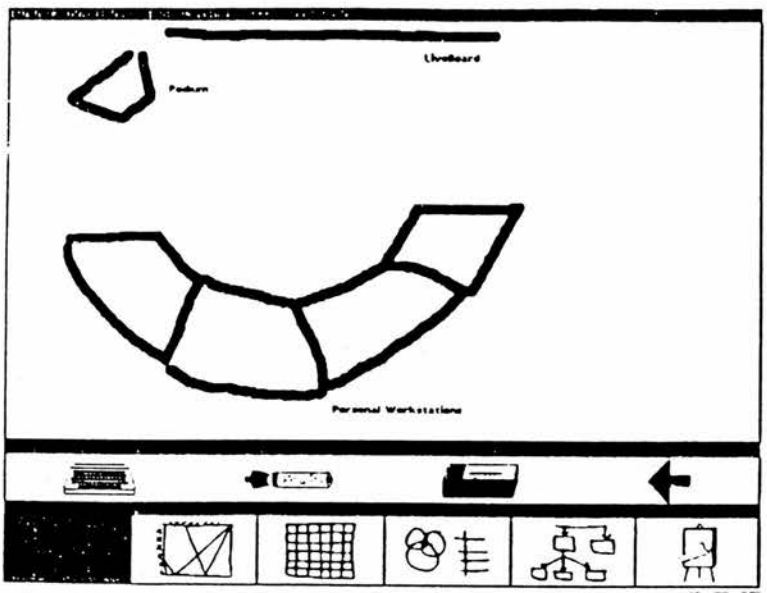


Fig. 5. Boardnoter of Colab at Xerox PARC provides participants with mouse-driven cursors ('chalk'), operating at individual workstations but not visible on the large meeting room screen. [Source: Fig. 13.2 of (Stefik et. al. 1987)]

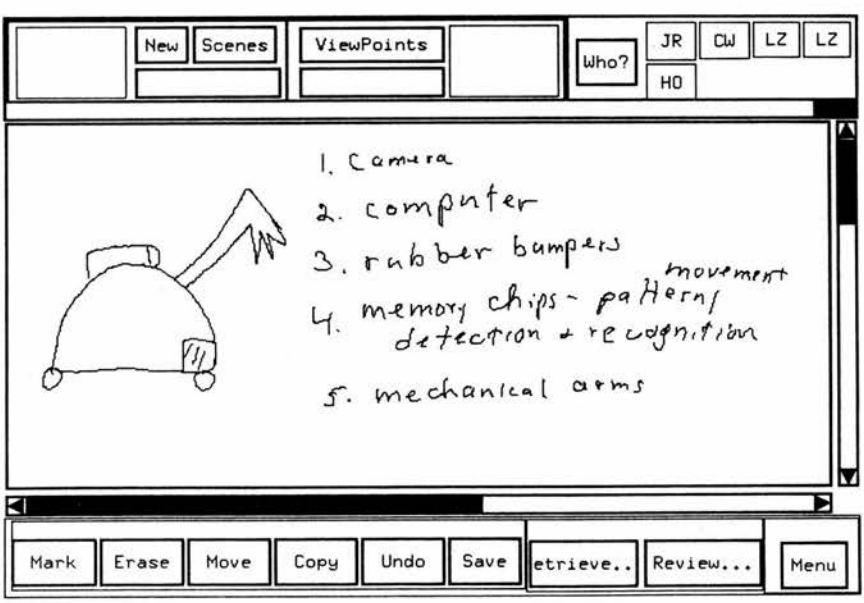


Fig. 6. Being similar to Boardnoter as a meeting suport tool, the We-Met drawing surface developed at IBM Watson Research Center has the feature of 'pen-based' interface. [Source: Fig. 1 of (Wolf et al. 1992)]

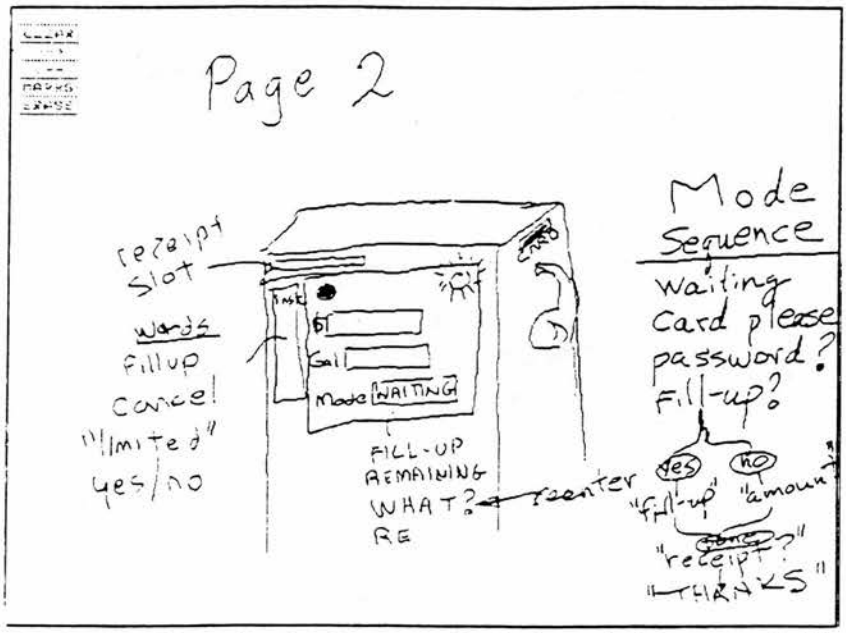


Fig. 7. The drawing/writing surface of the Commune workstation is comprised of transparent digitising tablets with styli; each digitizer tablet continually reports the position of its local stylus to the processor, and each user's stylus is represented on the screen as a pencil-shaped cursor producing marks in a distinct colour. [Source: Fig. 4 of (Minneman and Bly 1991)]

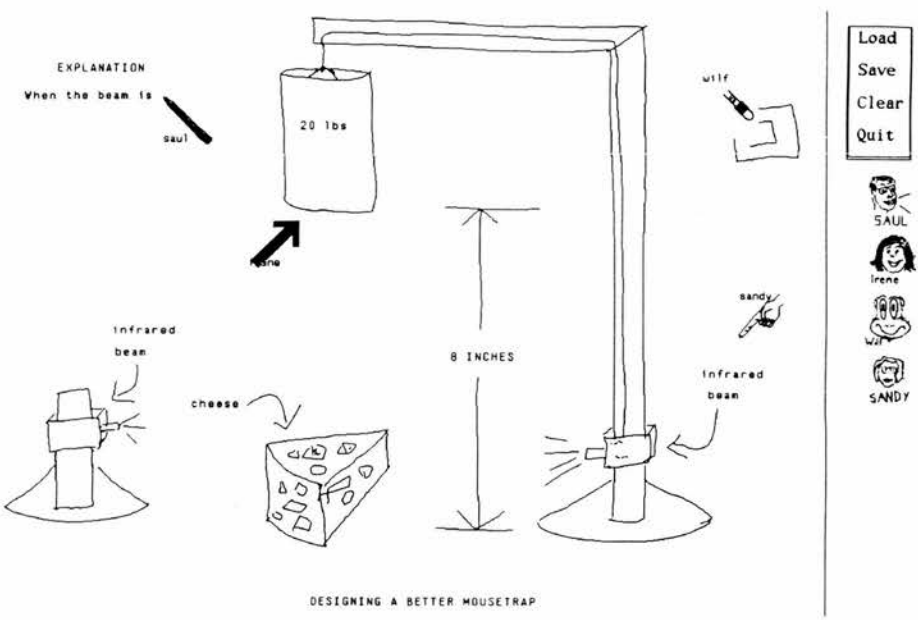


Fig. 8. In GroupSketch, multiple mouse-driven cursors represented by different icons are used to convey participants' physical gestures such as drawing, typing, pointing, erasing, and directing public attention. The positions and movements of the cursors at all sites are visible to all participants in real-time. [Source: Fig. 1 of (Greenberg and Bohnet 1991)]

etc. Due to its simplicity, pixel-based graphics has been used in several prototypes of shared drawing space. Operations like erasing, selecting-and-dragging, and cutting-and-pasting are commonly used for changing bits of drawn or textual expressions. Among the shared drawing tools built of pixel-based graphics, a diversity in the design of input devices shows different approaches to experimenting with how drawing events can be shared among participants (Figs. 5 to 8).

3. *object-structured graphics.* Freehand sketches and pixel-based graphics are unstructured graphical expressions to which very few operations can be applied. To be able manipulate parts of a drawings as the constructs of line, rectangle, circle etc., geometric structures need to be included in the implementation of graphics primitives. The term 'object-structured' as applied to graphics refers to a system's drawing primitives being programmed as *objects* and stored in a database to be addressed, manipulated, copied as individual entities. In an object-structured graphics system, types of drawing objects can modify themselves with various sorts of operations, such as creating, moving, resizing, grouping, rotating, duplicating, deleting etc. To give an example, one of the 'tool palettes' of Conversation Board (Brinck and Gomez 1992) provides a range of geometric objects including oval, line, arrow, and rectangles (see Fig. 9). Putting object-structured graphics into group use, there arises the

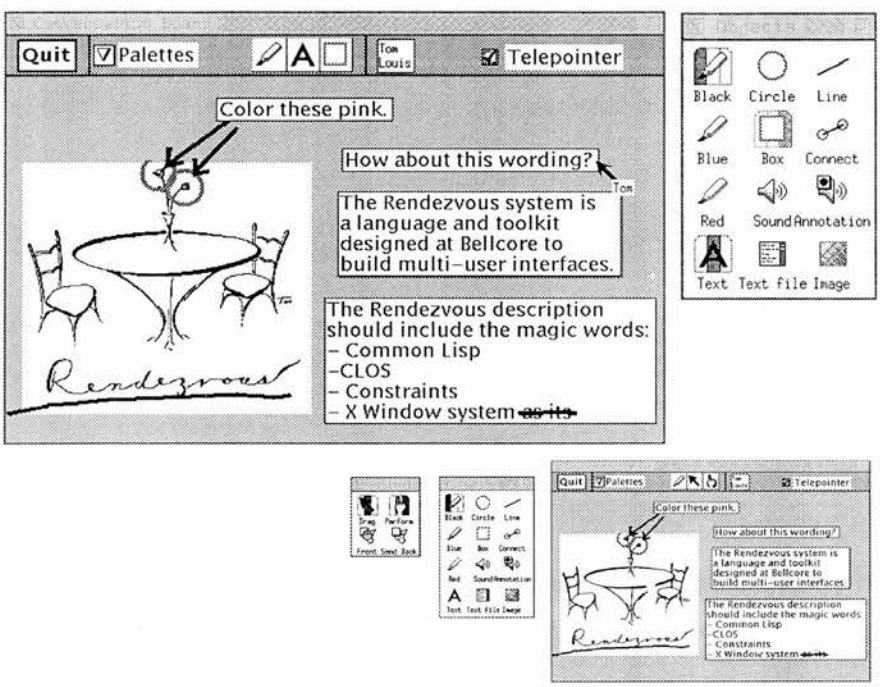


Fig. 9. The Conversation Board developed at Bellcore provides a number of structured objects including oval, line, arrow and rectangle; after objects are placed on the shared canvas they can still be edited and moved. [Source: Fig. 2 of (Brinck and Gomez 1992)] (Note that the 'Rendezvous' sketch shown here is an imported image, which was originally drawn by hand and digitised.)

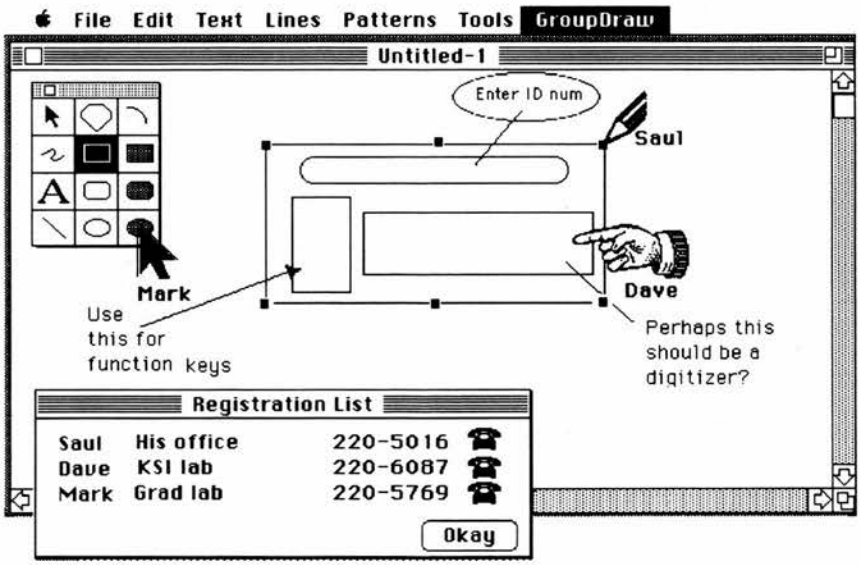


Fig. 10. GroupDraw was one of the first shared drawing systems using object-structured graphics to address the issue of concurrency control in collaborative drawing space. [Source: Fig. 2 of (Greenberg et al. 1992)]

problem of *concurrency control* which is not significant in pixel-based group graphics. In a session of collaborative drawing, it is likely that two or more designers intend simultaneously to manipulate the same object appearing on the shared drawing surface. To coordinate users' potential concurrent manipulations, the message-sending mechanism of object-oriented programming has been used in attempting to sequence concurrent processes at different sites. A good example is the design of object-structured group graphics in GroupDraw (Greenberg et al. 1991) (see Fig. 10). This example shows an interesting attempt to integrate the design of *communication primitives* with the design of the *graphics primitives*. As Greenberg's team did, two instance variables were built into the *root object* of all graphics primitives: *ownerProcess* and *coupling Status*.[8] By indicating who the owner of the process is, the former serves to arbitrate contention in manipulating an instantiated object; by indicating the status of object being 'private', 'public', or 'sharable', the latter determines the extent to which graphical objects are shared (Greenberg et al. 1991).

4. *knowledge-based graphics*. In contrast to manipulating objects at a syntactical level (as in object-structured systems), graphical objects are defined and manipulated *semantically* in knowledge-based graphics. In a knowledge-based approach, graphics primitives are programmed in terms of abstract construction and operation components that are specific to particular design domains. Evaluations, explanations, advices, alerts, or criticism of graphical expressions constructed in those components can then be computed and presented to

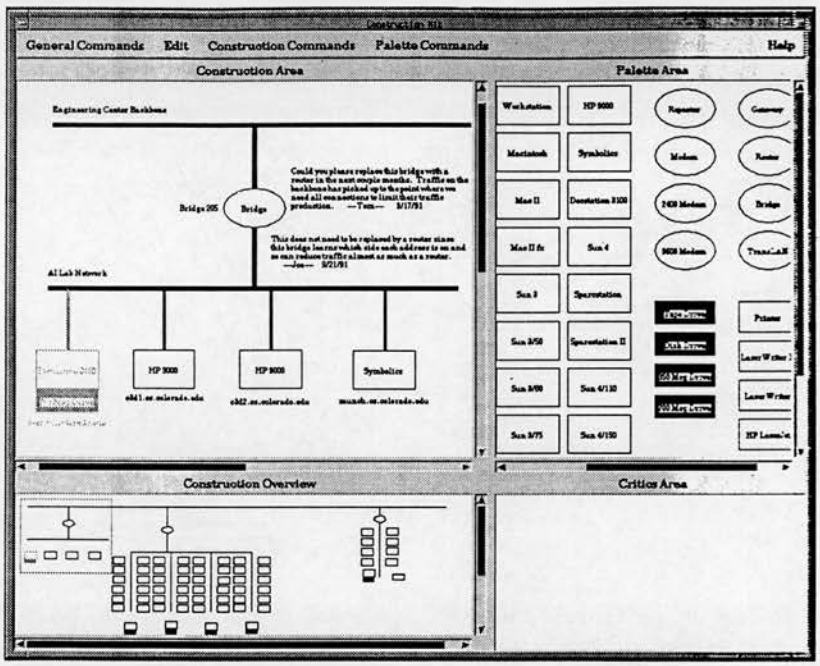


Fig. 11. The Construction Kit of XNETWORK which allows for a connection between graphical construction and a knowledge base representing 'group memory' of network design. [Source: Fig. 3 of (Reeves and Shipman 1992)]

the designer at work. A shared drawing space connected to a knowledge base provides a set of domain-specific graphic constructs as a common design language shared by its user groups. The drawing operations, which enable a user's direct manipulations of objects, are more conceptually bound to the system's knowledge domain; for example, parts of a construction can be manipulated by changing the values of attributes associated with the graphical constructs embedded. At a higher level, a user may gain *multiple views* of a design by switching from one underlying construction kit to another.[9] The design of XNETWORK (a recent update of NETWORK-HYDRA) environment is such an example (see Fig. 11), and it proposes a way of sharing drawing space through (indirect) collaborative construction of the common knowledge repository (Fischer et al. 1992; Revees et al. 1992).

5. *semi-structured graphics*. The term 'semi-structured' refers to a picture representation resulting from a mixture of unstructured graphics (freehand or pixel-based) with structured graphics (object-structured or knowledge-based). Currently, there appear two ways of enabling the use of semi-structured graphics in shared drawing space:

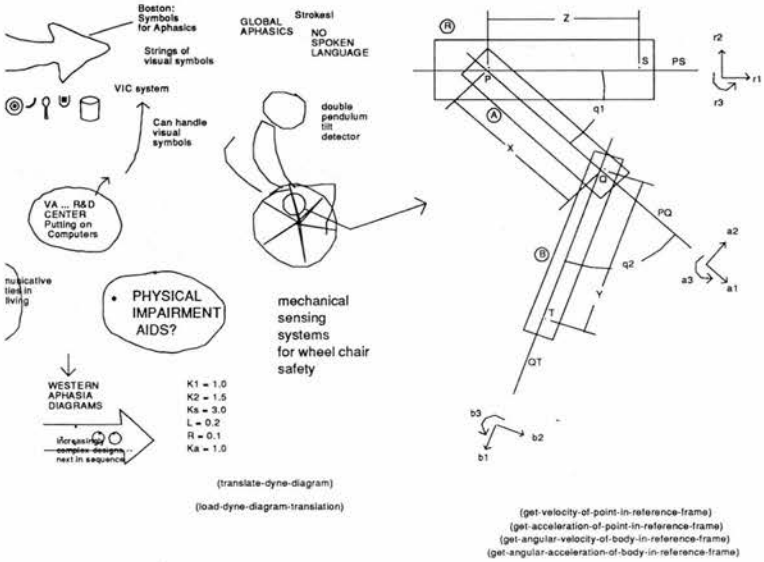


Fig. 12. Semi-structured graphics in the shared drawing space of vmacs enables both unstructured conversational expressions and visual language expressions which, as characterised by Lakin, are like 'coagulated lumps in oatmeal'. [Source: Fig. 17.7 of (Lakin 1990)]

- Formal drawings *embedded* in unstructured conversational sketches – the graphic editor vmacs[10] plus *visual languages for cooperation* is an example of this approach (Lakin 1990) (Fig. 12).
- Video captured images of freehand sketches *superimposed* on formal drawings generated by some graphics package – the design of TeamWorkStation presents a shared drawing space where transparent video images containing hand-drawn expressions are overlaid with formal drawings constructed on a computer screen (Ishii and Miyake 1991) (see Fig. 13).

3.2. COMMUNICATION NETWORKS AND INTERPROCESS COMMUNICATIONS

As shown by the observational studies, the different perspectives have led to various choices of what communication networks are appropriate for supporting the various patterns of shared drawing events. The term 'communication networks' has to cover a wider scope of system architectures in implementing shared drawing space; computer networks, as usually thought of, may not necessarily be involved here. Video networks, for example, have been exploited to support real-time collaborative drawing sessions held between remote working sites.

When computer networks are used to serve the communication infrastructure

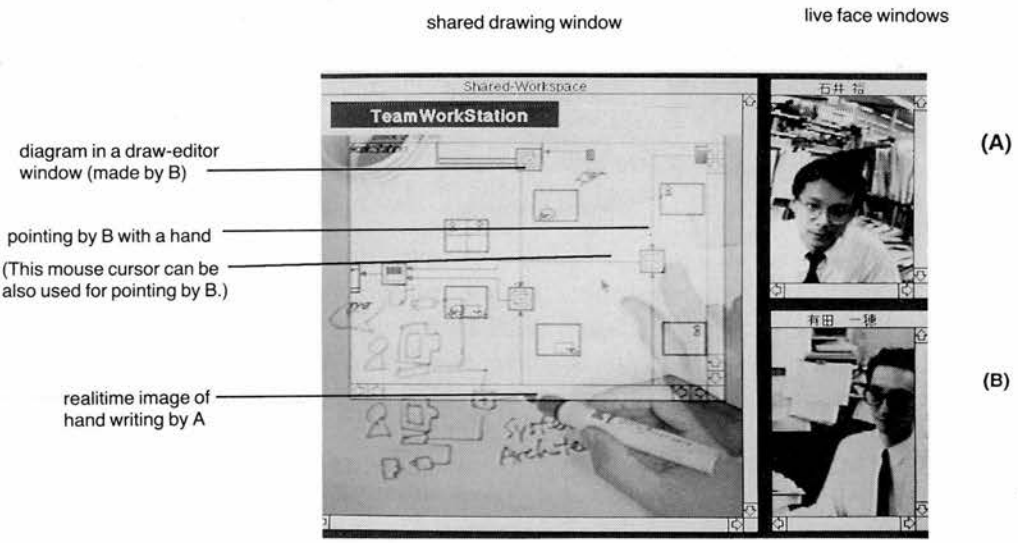


Fig. 13. The shared drawing space of TeamWorkStation facilitates a translucent overlay of a desktop image of a freehand sketch and an image of operational drawing, which is displayed on the shared screen of all participants for remote meetings. [Source: Fig. 5 of (Ishii and Miyake 1991)]

of collaborative drawing surfaces, there arise the issues of concurrency control and maintaining consistency in data and view sharing. Message passing seems to be the most widely employed mechanism to handle interprocess communication which receives inputs from multiple users' drawing acts and delivers the computed end results to each participating site. It is possible to classify current network configurations in shared drawing space into the following four different types.

1. *hard-wired configuration.* This is the network design adopted by most video-based, or computer and video fusion approaches. The components of the network are purpose-built to perform the specific system functions as a shared drawing surface. In a fully video-based configuration, video cameras, monitors, and projectors are *hard-wired* for imaging, transmitting, and projecting the images of participants and the state of their work. There may be two main reasons why a hard-wired architecture is constructed: (a) to investigate how *telepresence* can be realistically supported, which is conditioned by whether the configuration can convey co-operative work together with participants' 'body language' (e.g., hand gestures, facial expressions, eye contacts etc.) in the course of a remote meeting; (b) to simulate the elements of a natural setting of freehand sketching.[11]
2. *centralised configuration.* A shared drawing tool with a centralised communication structure can be explained by the 'star' network topology, in which all workstations are connected via a single link to a central switching node

(Sloman 1987). Within the configuration, a central server runs a single application and conference process. The conference process handles most of the synchronisation and serialisation issues, and the application process computes output of drawing functions from input multiplexed by the conference process. Each user's workstation runs a participant process (a user interface client), providing low level interactive graphics primitives.[12] Two constructs of interprocess communication in the client/server model have been borrowed from distributed programming: Rendezvous and Remote Procedure Call (RPC).[13] The advantage of a centralised architecture is the relative ease of maintaining synchronous display among distributed participant sites. Apart from heavy network demands and being less robust in the face of network and host machine failure, a major problem of the centralised approach is that it does not support participants' sharing the processes of creating and using drawing expressions, which is an important requirement as specified in (Bly 1988; Tang 1991), since transmissions take place only after drawing acts are completed.

3. *replicated configuration.* In contrast to the centralised approach, a replicated architecture runs a copy of the conferencing and application processes on every workstation that a user may interact with. The conference process at each site sends/receives input to/from other networked sites, and passes received input to the application process for generating output and

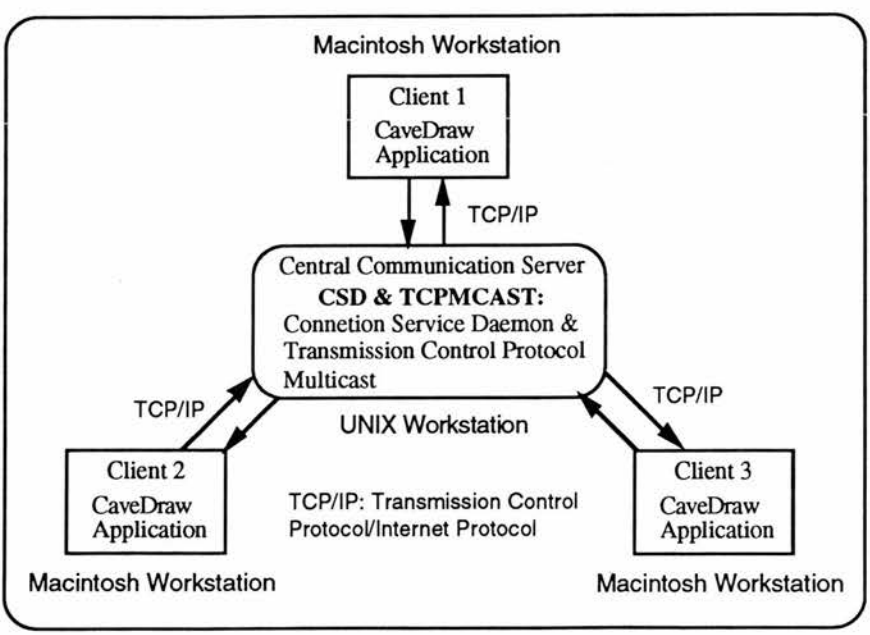


Fig. 14. The communication structure of CaveDraw an example of a hybrid network configuration. [After Fig. 17 of (Lu 1992)]

updating its resident display. Besides the merit of reduced network demands and hence gaining lower latency of application response on each local site, the replicated approach supports the conveying of drawing actions in the course of group meeting. A replicated configurations is built upon the *serial bus* or *highway* network topology, in which simultaneous transmission by multiple stations may result in interference; therefore, a media access control mechanism is needed to prevent or resolve contention for the transmission medium (Sloman 1987). As a consequence, it is harder for a replicated architecture to achieve integrity of shared drawing surfaces crossing all participating sites.

4. *hybrid configuration.* Here, a participant process run on every workstation may use a central conferencing process only for serialisation and synchronisation, and all other shared drawing space acts are communicated directly between participant processes. Taking CaveDraw as an example, a hybrid configuration may consist of a central communication server that mediates the participating workstations which all run a copy of the application program (Lu 1992) (Fig. 14). According to the CaveDraw experience, there remains the problem of concurrency because of the seemingly unavoidable time discrepancy between passing local events (by any one participant process) to the central communication manager and updating local displays (by the central manager).

3.3. INFORMATION STORAGE AND RETRIEVAL

To complete a shared design project, it may take participants hours, days, or even years to carry out individual as well as group tasks. The third class of design issues in shared drawing space involves the facilities to record and to manage the history of collaboration. Drawings and other forms of information created and used in the past may need to be brought back to the present for individual as well as group purposes. Several notions and functions of storing and retrieving information for supporting group design activity have been attempted.

1. *camera or video record.* To make records of work results from collaborative drawing sessions, fully video-base workspaces often resort to videotaping and/or picture-taking. Since the video-based approach puts great emphasis on the sharing of drawing actions rather than in data sharing, storage and retrieval have not been considered an essential function of a shared drawing surface. This decision is based on two explicit design rationales: that "drawings as artefacts in themselves are often meaningless" (Tang 1989); and that "hand gestures help participants to store (to remember) design discourse" (Tang 1991).
2. *workstation drawing files.* Workstation-based group drawing tools can make direct use of the file management facilities of most operating systems. In sup-

porting storing/retrieving data during collaborative drawing and design sessions, there appear to be different metaphors of storage and retrieval which lead to interfaces with various design features:

- (time-stamped) *pages or scenes*: In systems like GroupSketch, Commune, vmacs, or We-Met, the interface for storage/retrieval simulates the pages of a note pad or flip chart. New 'pages' or 'scenes' can be continuously issued for making marks; and, when facilitated by a history mechanism, one or more pages can be reloaded onto the current working surface.
- (time-stamped) *miniature sheets*. In Boardnoter, sketches are saved into reduced views as a collection of miniature sheets visible on the common screen. When retrieved, each sheet can be re-displayed at full screen size (Sefik et al. 1987).
- *drawing layers*. In CaveDraw (Lu 1992), the use of drawing files by a team of designers comes close to that of tracing paper by a design team, which facilitates transparent overlay drafting (Woods 1987). Drawings sketched on one or more layers can be stored in a single file; when recalled, it can be displayed together with other resident layers on one's drawing surface.
- *catalogue items*. In NETWORK-HYDRA, design representation is saved in a *catalogue* serving as a repository of designs constructed by participants over, perhaps, a long period of time. The catalogue contains several items, dealing with different aspects of the design task such as graphical construction, design rationales, design specification etc. Existing items in the catalogue can be accessed and copied into a new construction by modifying what is retrieved; and completed instances of design can be archived into the catalogue for future use or reference (Fischer et al. 1992).
- *content-directed retrieval*. As one of the visual languages designed for cooperation, Lakin proposed a novel function of retrieving drawing files by their contents period. It is suggested that content-directed retrieval might proceed by having users write and draw their expressions in a formal visual language, such as TEXT-GRAPHIC-QUERY (Lakin 1990).[14]

3.4. MULTI-USER INTERFACES

In supporting multi-party synchronous graphics interaction between collocated or remotely separated designers, there arise several novel design issues which are not common in traditional single-user drawing systems.

1. *telepointer*. A telepointer is a large cursor that appears on a common screen of a meeting room or on every workstation connected over a local network. As an interface device for group interaction, it is designed to be manipulated by participants (one at a time) to point to specific locations on a shared drawing

surface. Telepointing is often said to simulate the conveying of information by hand gesturing in group drawing activity. Limited effects of telepointing have been reported in, for example, Boardnoter and Conversation Board (see Figs. 5 and 9 respectively).

2. *multiple cursors.* The rationale underlying the design of multiple cursors is that multiple *identities* can be attributed to local cursors used by individual participants. Identity of a cursor can be defined by, for instance, the colour it produces, the name of its (current) user, or the gesture indicator it serves (i.e., pen, marker, eraser, pointer, etc.). But there is a tradeoff between the support for the various modes of gesturing and the support for rapid switching among drawing, writing and other actions (Bly 1988; Greenberg and Bohnet 1991; Brinck and Goemz 1992).
3. *group vs. individual views.* A shared drawing space may allow users to have different views that are local to individuals. On this issue, there appear different approaches to user-controllable view sharing:

   - A view sharing facility based on the "What You See Is What I See" (WYSIWIS) principle serves all participants' sharing strictly the same view during collaborative sessions; events taking place on any one site immediately affect the current states of the shared drawing surfaces appeared on other sites. In this case, no individual views are allowed for private work.
   - The WYSIWIS principle is relaxed to some extent so that the effects of manipulating parts of a shared drawing space can be kept locally. A participant can turn away from a public domain by scrolling the shared drawing interface to different places (e.g., in We-Met, see Fig. 6; and in GroupDraw, see Fig. 10), or by moving onto another drawing layer (e.g., in CaveDraw, see Fig.15).

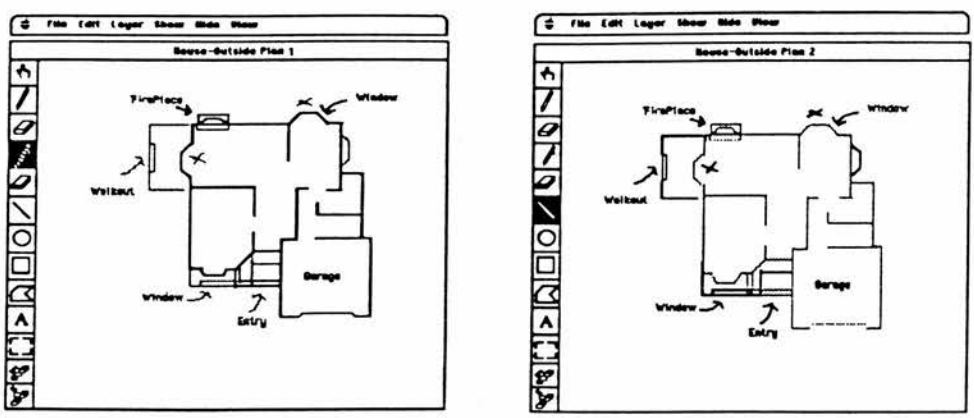


Fig. 15. The overlapping layered approach in CaveDraw: Two participants can have individual drawing surfaces when co-working on different layers. [Source: Figs. 12a and 12b of (Lu 1992)]

- A participant's workspace consists of an *individual screen* and a *shared screen*. With this workspace setup, a user is given great flexibility of controlling what and when things are to be made individual or public. Given the view sharing design of TeamWorkStation, a participant can simply drag a drawing or a document image on his or her own individual screen onto the neighbouring screen which can be synchronously shared by all networked working sites (Ishii and Miyake 1991).

3.5. OTHER DIALOGUE FACILITIES

In supporting rich and complex collaborative design activity, communication channels other than a shared drawing surface are often provided in parallel. At least three other dialogue channels have been regularly used to complement the design of shared drawing space:

- *Audio-links* are most widely employed to support participants' talking about parts of sketches while drawing or pointing at them.
- *Video links* are used not only to capture desktop images but also to transmit facial images of participants. There is a difference in the extent to which captured facial images are integrated with a shared drawing surface, and these are separate (e.g., in VideoDraw, Commune), juxtaposed (e.g., in TeamWorkStation), and entirely fused (e.g., in ClearBoard)
- A *messaging service* is provided in connection with a shared drawing surface, enabling users to send textual or graphics messages in the course of collaboration. The MUSK system was designed with a text-based messaging service (Crampton 1987); the visual language VISUAL-MAIL[15] working together with vmacs enables users to mail text-graphic pages (Lakin 1990). A mailing service is a practical alternative channel when audio/video links are not made available, or collaborative work is not based on synchronous interaction.

4. Shared drawing spaces in three group uses

In the above, we have reviewed the empirical or conceptual studies of group drawing activity and the design issues in developing prototypes of shared drawing support tools. To interrelate the two parts of the survey, this section presents a discussion of the use of the prototype tools developed. Seen in a broad view, it seems to us that the current research and development of shared drawing support tools aims to serve three different *group uses* of shared drawing tools: *conversation*, *management*, and *meeting*. Each use purports a way of collaborative working and thus a set of goals of system support. This view leads us to a grouping of the current prototypes into three categories:

1. *as conversation media.* As seen, most of the prototype designs are concerned with supporting real-time drawing interaction among remotely located participants who, presumably, come from more or less the same professional or technical background. To be used by group members, the functions of shared drawing space are devised mainly for supporting *conversations*. Systems built for the purpose may employ graphics primitives and drawing operations ranging from freehand sketches to object-structured graphics. Since facilitating *telepresence* is a major goal, the provision of multi-modal dialogue channels is as important as the setup of a shared drawing surface. Concerning shared use of the data resulting from conversations, however, current conversation-oriented systems provide comparatively less functionality for storing and retrieving during collaborative work sessions.
2. *as management media.* In the attempts to support collaboration via graphical and/or knowledge representations of design ideas or rationales, a shared drawing system may be characterised as a medium for users to *manage* collaborative design work. Two different views of supporting users' management behaviours have been expressed. As in the case of CaveDraw, one addresses the need for direct communication among participants; this view considers the users' need to organise design ideas that are often represented in graphical form. The other addresses indirect and long-term collaboration, considering the need of representing design knowledge in textual form; the formally captured knowledge is further connected to the system's drawing space. The shared drawing surface, as in the case of XNETWORK, is shared in the sense that the graphics construction space has formal (internal) communication with the state of a shared knowledge base containing design rationales written down by, perhaps, designers involved in the past.
3. *as performing media.* The third design perspective considers shared drawing space activity taking place in *meetings*, participated by two working sub-groups; namely, a single or multiple *performers* (or *demonstrators*, *facilitators*), and several *viewers* (or *commentators*). Group drawing or diagramming activities normally take place within a meeting room equipped with a conferencing system, as in the case of Boardnoter, where the performer has exclusive access to the big common screen and all other participants sitting as attendees during a brain-storming session. In We-Met, the strict WYSWIS feature is relaxed to the extent where participants can scroll a shared scene into private areas without affecting each other, but the change of a whole scene remains exclusively controlled by one individual. Concerning the choice of drawing primitives, the use of semi-structured graphics in vmacs is a novel idea, which seeks a compromise between the support of agility/openness by conversational graphics and that of expression processibility by computerised interpretations.

Based on the review of the system features and the grouping of the prototypes, an overview of the current survey is presented in Table 3. Given the fact that the research on collaborative drawing support tools is a continuously- (and rapidly-) expanding field, the above categorisation may be superficial. Nevertheless, to make a small contribution to the research in this field, it is hoped that this survey can serve, at least, as an index of what design issues have been considered.

5. Conclusions and pointers to future investigations

In this paper, drawing on a selection of observational studies and research prototypes, we have presented a survey of the present trends of CSCW research and designs in shared drawing space. This is a rapidly developing research area; during the writing of this paper fresh work will undoubtedly be done. But based on what we have studied, some remarks may be drawn on the nature of researching and developing system functions and architecture of shared drawing space. Basically, how do we justify the attempts that have been made, and can we suggest some pointers to further investigations?

In our overall discussions, distinctions between *drawing* and *designing* have not been made. Based on the differences observed between simple electronic sketch pads for people to draw together, and rather sophisticated systems for cooperative designers to develop and manage ideas graphically, we may point out that drawing and designing cover somewhat different aspects of human activity, which give rise to the diversity in thinking about what is to be supported:

- Drawing can be considered as a general form of human communication in which pictures and texts are the traces left by communicative acts which took place sometime in history. Shared perception and understanding of the traces, thus communication, is highly conditioned by the sharing of drawing actions that produce and use the physical traces.
- Designing seems to centre around some previously or currently existing, or yet to exist, *design artefacts*, which does not necessarily involve drawing activity; but in some design fields there can be no designing without drawing. Structures of design artefacts are introduced by, or emerge from, people's design thinking, which often conditions communication and collaboration among participating designers.

Systems that support group drawing activity must deal with the issues of supporting direct communications among users who may not be present face-to-face. Unstructured graphics prove to be a good choice for a shared drawing surface responsive to demands for (a) the intimacy between what is drawn and the actions of drawing, (b) the speed needed to maintain conversations, and (c) the freedom of making expressions. Structured graphics has been attempted in group

Table 3. A categorisation of the current research on collaborative drawing support tools in terms of different group uses and system features.

SYSTEM FEATURES \ RESEARCH PROTOTYPES		Conversational Mediums								Management Mediums		Performance Mediums		
		Video-White-Board	Com-mune	Team-Work-Station	Group-Sketch	X-sketch	Group-Draw	Clear-Board - 2	C'nversation Board	Cave-Draw	NET-WORK-HYDRA	Board-noter	vmacs Visual Langs.	We-Met
Graphics Primitives & Drawing Operations	direct drawing	●	◍	◍	○	○	○	○	○	○	○	○	○	◍
	pixel-based	○	●	◍	●	○	○	●	○	●	○	●	●	●
	object-structured	○	○	◍	○	●	●	◍	●	○	○	○	○	○
	knowledge -based	○	○	◍	○	○	○	○	○	○	●	○	●	○
	semi-structured	○	○	●	○	○	○	○	●	○	○	○	●	○
Communication Networks	hard-wired	●	●	◍	○	○	○	●	○	○	○	◍	◍	○
	centralised	○	◍	○	◍	●	○	○	●	◍	●	●	●	○
	replicated	○	○	○	●	◍	●	●	○	◍	○	○	○	●
	hybrid	○	◍	◍	○	◍	○	○	○	●	○	◍	◍	○
Information Storage and Retrieval	cameras/ videos	●	●	●	○	○	○	●	○	○	○	○	○	○
workstation drawing files	miniature	○	○	○	○	○	○	●	○	○	○	●	○	○
	pages	○	●	◍	●	●	●	○	●	○	○	◍	●	●
	layers	○	○	◍	○	○	○	●	○	●	◍	○	○	○
	catalog	○	○	○	○	○	○	○	○	○	●	○	◍	○
	patterns	○	○	○	○	○	○	○	○	○	○	○	●	○
Multi-User Interfaces Design	telepointer	○	●	◍	○	●	○	○	●	●	○	●	○	○
	multiple cursors	○	●	●	●	○	●	●	○	●	○	●	○	○
	WYSIWIS	●	●	○	●	●	○	○	○	○	○	●	●	○
	relaxed WYSWIS	○	○	○	○	○	●	●	●	●	○	○	○	●
	separate screens	○	○	●	○	○	●	○	○	○	○	◍	◍	◍
Other Dialog Channels	audio links	●	●	●	○	○	○	●	◍	○	○	○	○	○
	video links	●	●	●	○	○	○	●	◍	○	○	○	○	○
	messaging	○	○	○	○	○	○	○	○	○	◍	○	●	○

LEGEND: ○ Not Developed or Used ◍ Developed or Used to a Degree ● Developed or Used

drawing mainly for increasing the functionality of drawing surfaces and for concurrency control; but its real effectiveness in supporting synchronous graphics communications remains to be demonstrated.

Systems that support group work in design need to provide shared drawing space for construction and communication. The problem of how to integrate formal or informal representations of graphical objects and design ideas with an understanding of communication and coordination in design remains to be explored more deeply. We have seen two alternative approaches to group design support systems: structured construction with argumentation and semi-structured conversation with interpretation. Yet, regarding the former, we have not seen approaches that support group design processes based upon synchronous and heterogeneous representations and uses of design knowledge. In respect of the latter, the concept of supporting *autonomy* in addition to that of *heterogeneity* may need to be further addressed such that (a) organisationally, there is less discrepancy between individuals' making contributions and gaining benefits or satisfaction through the use of technologies, and (b) administratively, teamwork can be freed from one single managerial ambit. As a general direction, we have to consider what is essential to design practices that normally demand *interdisciplinary* participation, as well as *integrated* design products.

Drawing, being basic to design, is a universal mode of communication in numerous fields of human cooperative endeavour, and the research pursued in shared drawing space has indeed been world-wide. It can be justified that the understanding and developments made or yet to be made in this research area can make its own contributions to the advancement of CSCW systems. This can be reinforced by recalling the basic problems raised by the CSCW research community – the aspects of *group processes*. Among many others, Paul Wilson has outlined four aspects of group processes basic to the theory, practice, and design of CSCW (Wilson 1991). By referring to, particularly, the first two aspects, we would like to point out that research on collaborative drawing and design present interesting issues that are worth further investigating:

1. *individual work patterns*. The design of group support tools and working practices has to take into account individual work habits and predilections. In this aspect, design activities present a high degree of idiosyncrasies among participants. How can users specify personal constructs or tools as prerequisites for supporting individual work patterns? This consideration opens up potential system design issues concerning (a) participants' *deriving* individual design spaces from the design space they share, and (b) how a common design space may *emerge* and *evolve* from the interaction between individual design spaces.
2. *representation of organisational knowledge*. Research on structures of graphics can be further pursued to develop the representation of *organisational knowledge* graphically [see, for example, (Star 1989)]. Can a simple graphical

approach contribute to a better management of organisational knowledge which by its nature tends to be difficult to locate, recall, and update?

Acknowledgements

COPYRIGHT ACKNOWLEDGEMENTS

The following illustrations are published by kind permission of the following: Figure 2 and Fig. 3 are from John Tang, © 1991 by the ACM, Inc. Figure 4 and Fig 13 are from Hiroshi Ishii, © 1992 and © 1991, respectively, by the NTT Human Interface Labs. Figure 5 is © 1987 by the ACM, Inc. Figure 6 is from Catherine Wolf, © 1992 by the ACM, Inc. Figure 7 is from Sara Bly, © 1991 by the ACM, Inc. Figure 8 is from Saul Greenberg, © 1991 by Morgan Kaufmann, Inc. Figure 9 is from Tom Brinck, © 1992 by the ACM, Inc. Figure 10 is from Saul Greenberg, © 1992 by Butterworth-Heinemann, Inc. Figure 11 is © 1992 by the ACM, Inc. Figure 12 is from Fred Lakin, © 1990 by Lawrence Erlbaum Associates. Figure 15 is from Iva Lu, © 1991 by Kluwer Academic Publishers.

PERSONAL ACKNOWLEDGEMENTS

The author wishes to thank Sara Bly, Tom Brinck, Gerhard Fischer, Saul Greenberg, Hiroshi Ishii, Fred Lakin, Jeff Lee, Iva Lu, Marilyn Mantei, John Tang, and Catherine Wolf for providing him with essential research articles, technical reports, and e-mail discussions. Thanks to John R. Lee, Aart Bijl at EdCAAD, for their initial comments on this paper, and to the anonymous referees, for their constructive comments upon which the author's revision was based. This work was carried out when the author was partically supported by the UK ORS Awards Scheme.

References

Bal, H.E., Steiner, J.G. and Tanenbaum, A.S. September, 1989. Programming Languages for Distributed Computing Systems. *ACM Computing Survey*, 21(3): 261–322.

Bly, S.L. and Minneman, S.L. 1989. Commune: A Shared Drawing Surface. Technical Report SSL-89–86, System Sciences Laboratory, Palo Alto Research Center.

Bly, S.L. 1988. A Use of Drawing Surfaces in Different Collaborative Settings. In ed. I., Greif, *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW' 88)*, 250–256, ACM Press.

Brinck, T. and Gomez, L.M. 1992. A Collaborative Medium for the Support of Conversational Props. In (eds) *Proceedings of CSCW' 92*, Turner, J. and Kraut, R. 171–178. New York: ACM Press.

Brink, T. 1993. Building Shared Graphical Editor Using the Abstraction-Link View Architecture. To appear in *Proceedings of ECSCW'93* (European Conference on Computer-Supported Cooperative Work, Milano, Italy, September, 1993).

Conklin, J and Begeman, M.L. October 1988. gIBIS: A Hypertext Tools for Exploratory Policy Discussion. *ACM Transaction on Office Information Systems*, 6(4): 303–331.

Crampton, C. 1987. MUSK a Multi-User Sketch Program. In *Proceedings of the European UNIX Systems User Group*, 17–29.

Dewan, P. and Choudhary, R. 1991a. Flexible User Interface Coupling in a Collaborative System. In *Proceedings of the ACM CHI'91 Conference*, 41–48. ACM Press.

Dewan, P. and Choudhary, R. 1991b. Primitives for Programming Multi-User Interfaces. In *Proceedings of the ACM Symposium on User Interfaced Software and Technology (UIST' 91)*, 69–78. ACM Press.

Ellis, C.A., Gibbs, S.J., and Rein, G.L. 1991. Groupware: Some Issues and Experiences. *Communications of the ACM* 34(1): 39–58.

Fischer, G., Lemke, A., Mastaglio, T. and I., Morch, A. April 1991. The Role of Critiquing in Cooperative Problem Solving. *ACM Transactions on Information Systems,* 9(3): 123–151.

Fischer, G., Grudin, J., Lemke, A., McCall, R., Ostwald, J., Reeves, B. and Shipman, F. 1992. Supporting Indirect, Collaborative Design with Integrated Knowledge-based Design Environments. *Human Computer Interaction* 7(3): 281–314.

Greenberg, S. and Bohnet, R. 1991. GroupSketch: A Multi-user Sketchpad for Geographically-distributed Small Groups. In *Proceedings of Graphics Interface '91*, pp. 207–215, Calgary, Alberta, June 5–7, Morgan Kaufmann.

Greenberg, S. July 1991. An Annotated Bibliography of Computer Supported Cooperative Work. *SIGCHI Bulletin* 23(3): 29–62.

Greenberg, S., Roseman, M., Webster, D. and Bohnet, R. July 1992. Human and Technical Factors of Distributed Group Drawing Tools. Interacting with Computers, 4(1), pp. 364–392, December. Butterworth-Heinemann.

Ishii, H. 1990. TeamWorkStation: Towards a Seamless Shared Workspace. In *Proceedings of CSCW'90*, 13–26.

Ishii, H. and Ohkubo, M. 1990. Design of TeamWorkStation: A Realtime Shared Workspace Fusing Desktops and Computer Screen. In *Proceedings of IFIP WG8.4 Conference on Multi-User Interfaces and Applications*, eds. S. Gibbs and A.A., Verrighn-Stuart. North Holland.

Ishii, H. and Arita, K. 1991. ClearFace: Translucent Multiuser Interface for TeamWorkStation. In *Proceedings of ECSCW' 91*, eds. L. Bannon, M. Robinson and K. Schmit, 163–174.

Ishii, H. and Miyake, N. December 1991. Toward an Open Shared Workspace: Computer and Video Fusion Approach of TeamWorkStation. Communications *of the ACM* 34 (12): 37–50.

Ishii, H. and Kobayashi, M. 1992. ClearBoard: A Seamless Media for Shared Drawing and Conversation with Eye-Contact. In *Proceedings of CHI'92*, 525–532, ACM Press.

Ishii. H., Kobayashi, M. and Grudin, J. 1992. Integration of Inter-Personal Space and Shared Workspace: ClearBoard Design and Experiments. In *Proceedings of CSCW'92*, eds. J. Turner and R. Kraut, 33–42, New York: ACM Press.

Kunz, W. and Rittel, H.W.J. 1970. Issues as Elements of Information Systems. Technical Report 131, Institute of Urban and Regional Development, University of California, Berkeley, California.

Lakin, F. 1983. Measuring Text-graphic Activity. In *Proceedings of the GRAPHICS INTERFACE'83*, Edmonton, Alberta.

Lakin, F. 1986. Spatial Parsing for Visual Languages. In eds. S.K. Chang, T. Ichikawa and A.P. Ligomenides. *Visual languages*, 35–85. Plenum Press.

Lakin, F. 1988. A Performing Medium for Working Group Graphics. In *Computer Supported Cooperative Work: A Book of Readings*, ed. I. Greif, 366–396. Morgan Kaufman Publishers.

Lakin, F. 1990. Visual Languages for Cooperation: A Performing Medium Approach to Systems for Cooperative Work. In eds. J. Galegher, R.E. Kraut and C. Egido. *Intellectual TeamWork: Social and Technological Foundations of Cooperative Work*, 453–488. Lawrence Erlbaum Associates.

Lu, I.M. and Mantei, M. 1991. Idea Management in a Shared Drawing Tool. In eds. *Proceedings of ECSCW'91*, L. Bannon, M. Robinson and K. Schmit, 97–112. Kluwer Academic Publishers.

Lu, I.M. 1992. Supporting Idea Management in a Shared Drawing Tool. Unpublished Master Thesis, Department of Computer Science, University of Toronto.

Minneman, S.L. and Bly, S.A. 1991. Managing á Trois: A Study of a Multi-user Drawing Tool in Distributed Design Work. In *Proceedings of the Conference on Human Factors in Computing Systems*, 217–224. ACM Press.

Patterson, J.F. 1991. Comparing the Programming Demands of Single-user and Multi-user Applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'91)*, 87–94.

Reeves, B. and Shipman, F. 1992. Supporting Communication between Designers with Artifact-Centered Evolving Information Spaces. In eds. *Proceedings of CSCW'92*, J. Turner and R. Kraut, 394–401. New York: ACM Press.

Rein. G.L., and Ellis, C.A. 1991. rIBIS: A Real-time Group Hypertext Systems. In *International Journal of Man-Machine Studies* (34): 349–367. Also in *Computer-supported Cooperative Work and Groupware*, ed. S. Greenberg, 223–242. London: Academic Press.

Rodden, T. December 1991. A Survey of CSCW Systems. *Interacting with Computers* 3(3): 319–353.

Schön, D. 1985. *The Design Studio: An Exploration of its Tradition and Potentials*, 30–52. RIBA Publications Ltd.

Silverman, B.G. April 1992. Survey of Expert Critiquing Systems. Practical and Theoretical Frontiers. *Communications of the ACM* 35(4): 106–127.

Sloman, M. and Kramer, J. 1987. *Distributed Systems and Computer Network*, 128–129. Prentice-Hall.

Star, S.L. 1989. The Structure of Ill-structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving. In *Distributed Artificial Intelligence Vol. 2 (Research Notes in Artificial Intelligence)*, eds. Les Gasser and M.N. Huhns, 37–53. London: Pitman.

Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S. and Suchman, L. 1987. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM* 30(1): 32–47.

Tang, J.C. and Leifer, L.J. 1988. A Framework for Understanding the Workspace Activity of Design Teams. In *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88)*, 26–28. ACM Press.

Tang, J.C. and Minneman, S.L. April 1991a. VideoDraw: A Video Interface for Collaborative Drawing. *ACM Transactions on Information Systems* 9(2): 170–184.

Tang, J.C. and Minneman, S.L. 1991b. VideoWhiteBoard: Video Shadows to Support Remote Collaboration. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, 315–322. ACM Press.

Tang, J.C. 1989. Listing, Drawing, and Gesturing in Design: A Study of the Use of Shared Workspace by Design Teams. Technical Report SSL-89–3, Palo Alto Research Center, CA.

Tang J.C. 1991. Findings from Observational Studies of Collaborative Work. *International Journal of Man Machine Studies* 34: 143–160.

Tatar, D.G., Foster, G., and Bobrow, D. 1991. Design for Conversation: Lessons from Cognoter. In ed. S. Greenberg. *Computer-supported Cooperative Work and Groupware*. 55–80. London: Academic Press.

Vergopoulos, S. 1991. A Multi-layered Model for Interpreting Design Drawings. In *Proceedings of the EuroplA 91*, 150–167.

Wilson, P. 1991. *Computer Supported Cooperative Work: An Introduction*. Dordrecht: Oxford Intellect, Kluwer Academic Publishers.

Wolf, C.G., Rhyne, J.R. and Zorman, L.A. 1991. WE-MET (Window Environment-Meeting Enhancement Tools). *CHI'91 Proceedings*, 441–442.

Wolf, C.G., Rhyne, J.R. and Briggs, L.K. 1992. Communication and Information Retrieval with a Pen-based Meeting Support Tool. In eds. *Proceedings of CSCW'92* J. Turner and R. Kraut, 322–329. New York: ACM Press.

Woods, F. and Powell. J. 1987. *Overlay Drafting: A Primer for the Building Design Team*. London: Architectural Press.

Notes

1. It may be controversial to include the observations and systems made by Fischer's group in our survey. However, the inclusion is intended to establish a wider concept of 'shared drawing' activities. In particular, as mentioned in the beginning of the paper, we fell the need for an examination of any possible links between supporting shared drawing activities and supporting collaborative design. To us, Fischer's systems present a distinct attempt to support a form of shared drawing activities, involving participants' indirect (i.e., asynchronous) communication for exchanging technical knowledge. It is evident that, in Fischer's

systems, 'graphical construction' is an integral part of the overall collaborative design processes, though it may appear highly task-oriented.

2. Similar time space matrices have been proposed by Johansen (Johansen 1988) and Ellis and others (Ellis et al. 1991) in their discussions of groupware design.
3. To this point, 'collaborative' seems not an obvious factor for 'collocated asynchronous' being fundamentally different from 'remote asynchronous' if the actual physical distance is taken into account. But, basically, 'collocated' implies that members can see and talk to each other without moving from one place to another. To better illustrate this event type, the author wishes to quote the comments given by one of the anonymous referees of the paper by referring this to 'team room' activities, and another good example of collocated asynchronous is 'shift work' (i.e., when one shift passes information to the next shift).
4. The argument here refers to the argumentation in the Issue Based Information System (IBIS) method originally developed by Kunz and Rittel (Kunz and Rittel 1970), and extended by Conklin and Begeman in the gIBIS tool (Conklin and Begeman 1988). According to Fischer et al., the issue-based argumentation method is intended as an interpretation of the 'reflection' in the design theory of *reflection-in-action* proposed by Schön (Schön 1985).
5. As a method parallel to argumentation, critiquing (the generation and sending *critic messages*) is developed by the Fischer group to identify and explain why a given design construction is inconsistent with the state of group memory (the so called breakdown situations). The critiquing method has been experimented by the same research group in implementing several cooperative problem solving systems (see Fischer et al. 1991, for more details). For the theory and practice of expert critiquing systems, a comprehensive survey can be found in (Silverman 1992).
6. For instance, an individual can cut parts of a drawing originally in his or her own colour marker and then paste them into the public domain in pencil marks (Lu and Mantei 1991).
7. The design of ClearBoard-1 was subsequently updated to ClearBoard-2 in which a multiuser paint editor, TeamPaint, run on network Macintosh computers was integrated with the original video network and drawing boards. For a detailed report on the design and use of the ClearBoard-2 system, see (Ishii, Kobayashi and Grudin 1992).
8. The concepts and algorithms of *flexible coupling* and *coupling awareness* were firstly explored and used in programming multi-user interfaces by Dewan and Choudhary [see (Dewan and Choudhary 1991a; Dewan and Choudhary 1991b) for more details].
9. Note that multiple views of a *design* is different from multiple views of a *drawing*. In a kitchen design, for example, multiple views such as structure, lighting, mechanical services etc., can be involved, and each view may produce drawings in a distinct domain of construction. Multiple views of, say, a sketch of a kitchen plan, on the other hand, require multiple *interpretations* of a single graphical construction from different views. For an exposition of a multi-layered model for interpreting architectural drawings, see (Stavros 1991).
10. vmacs is a trademark of the Performing Graphics Company.
11. For example, in the prototype design of Commune, Bly and Minneman have described the decision of modelling the system after a shared pad of paper, which leads to the use of a horizontally positioned drawing surface and a writing tool like a pen (Bly and Minneman 1989).
12. In Patterson's (Patterson 1991) terms, the centralised application and conference process is the single *abstraction process* containing the abstraction objects for the application; and a participant process is a *view process* containing the view objects for a particular user.
13. More detailed explanations of the difference between the two communication primitives can be found in (Bal et al. 1989).
14. TEXT-GRAPHIC-QUERY is a trademark of the Performing Graphics Company.
15. VISUAL-MAIL is a trademark of the Performing Graphics Company.

On the Emergence of Common Design Metaphors in Collaborative Design

Chengzhi Peng*
Information Technology Research Institute,
University of Brighton, Lewes Road,
Brighton, East Sussex BN2 4AT, UK.

Abstract

This paper reports an observation of how designers work together to achieve integrated building design on the basis of individual contributions. A case study shows that the keystone of fruitful collaboration lies in the projection of *common images*; and *common design metaphors* emerging from group interpretation of the common images can function as a communicative device, allowing for participants to collaborate effectively. A set of *constraints on collaboration* is identified via a situation-theoretical analysis of a scenario of collaborative design. As shown, the constraints derived have some implications for building collaboration-supporting tools.

1 An Architecture Dreaming about Fish?

In my earlier case studies of teamwork in architectural design, reported in [6, 7], one of the distinct approaches to teamwork in building design was characterised as 'metaphorist.' And one of the historical cases observed shows the following facts (See Figure 1):

> The Domo Serakaito, built in 1974, is a house christened "coelacanth" because of its (planar) shape, created within a long, heterogeneous group process: five members of the design team designed individual sections, which was allowed to show in the clear joint within the complete building. They gain unity from the image of fish.[1]

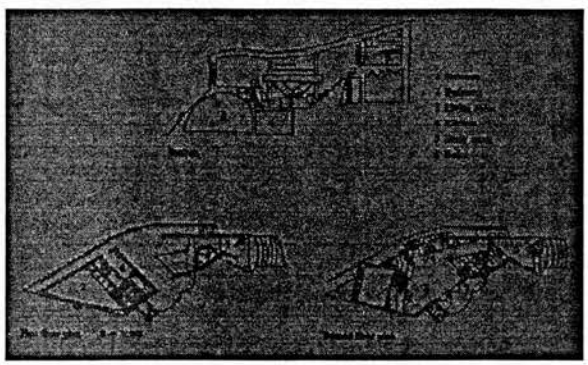


Figure 1: The plan of the Domo Serakanto Kamakura, designed by Team Zoo, Kanagawa Prefecture, Japan, 1974.

The above and other design literature suggests the existence of a distinct teamwork pattern for which we name it *metaphorist*. It involves collective projection and interpretation of 'common images' on the basis of individual parts contributed by different designers. Collaborative design of this pattern presents some interesting features that demand a systematic explanation which, in turn, may serve as references for future system design.

Consider the statement below, given by Spiedel in the same book [8, p.15], as a 'puzzle' to be solved:

> "Team Zoo has discovered that teamwork, even when markedly individual, can still produce a coherent whole, if a common image exists that each individual can interpret differently, and that allows a great deal of scope."

*Formerly, EdCAAD, Department of Architecture, University of Edinburgh, 20 Chambers Street, Edinburgh EH1 1JZ, UK.

[1] For this design project, Team Zoo's original statement read as "... What came out from almost three years of struggle was a coelacanth that crawled out from the sea. Domo Serakanto, with its gill, spine, horn, cilium, teeth, antennae and scales, it appeared in the wind and sank in the light. Domo Serakanto is a fish dreaming about architecture, an architecture dreaming about fish." [8, p.32].

This paper attempts to propose a descriptive theory of collaborative design with the metaphorist features briefly introduced above. The theory aims to explain what constitutes collaboration, and its implications for tool building will follow.

2 The Metaphorist Scenario: An Abstract

An overview of the metaphorist approach to collaborative design in the following simple (somehow abstract) scenario. The scenario also highlights some of the teamwork features to be focused upon. A more elaborate account of these features is given in the next section.

> At the inception of a design project, designers of different expertise and perspectives firstly set up individual workspaces for generating and modifying design expressions targeted at particular design aspects (domains). Participants' setting up individual workspaces may be distributed over several remote working sites without communication in the first instance.
>
> At some (later) design stage, participants take part in meetings to jointly present their local design decisions in a common workspace. On viewing the gathered design expressions, potential connections among individual works may be perceived and discussed among group members. Motivated by the joint conception of putting things together, participants proceed to carry out integration tasks collectively in the common workspace.
>
> By inspecting the resultant integration, participants are, individually, motivated to carry out further design developments in their own working domains. They may decide to modify or refine the design expressions previously made.
>
> As local design solutions are explored or elaborated to a certain extent, participants meet again. Regarding the individual design works newly arrived at, designers carry out, again, integration tasks to reach new states of integrated design.

The scenario given above is short indeed but sufficient to raise some questions:

- How can common design images be aggregated or projected collectively by participants if they know little about each other's design domain?
- It seems to be the case that aggregated design parts or projected overall design consequences may give rise to new developments in individual design works, and vice versa; how can we give an account of the apparently dual communication between what is integrated in a common workspace and what is currently developed in distributed individual workspaces?
- Given the sharing of common design images among participants, how do design changes made in one individual workspace affect those in others?

My current explanation toward the above questions comes from an analysis based on the situation-theoretical framework originally introduced by Barwise and Perry [3, 1]. In particular, by examining the constraints on the *flow of information* among different *situation types*, the current study produces a situation-theoretical account of the metaphorist approach to collaborative design.

3 A Situation-Theoretical Exposition

Given the scenario of teamwork in design abstracted above, it seems natural to stipulate that design is basically a *modelling* activity. By 'modelling', we refer to the performing of *modelling acts* (mental or otherwise) in *modelling spaces* (physical or abstract) by some individuals. Collaborative design, therefore, involves multiple threads of modelling activities that give rise to complex interactions. As will be shown in what follows, one of the benefits of seeing design as modelling is that we can thus arrive at a descriptive theory that unifies the representation and communication aspects of collaborative design.

Our presentation of the descriptive theory consists of four parts. We firstly outline the basic kinds of modelling acts and modelling spaces observed in the metaphorist scenario. By putting the action and the space aspects together, an *action-space* matrix can then be constructed, which specifies eight different *situation types* in collaborative design. Thirdly, we spell out the possible connections among the situation types identified, resulting in a map (or, a pattern) of the *flow of information* in the metaphorist approach. Finally, *constraints on collaboration* are searched out by examining the conditions for information to flow from one situation type to another.

3.1 Modelling Spaces

In seeing design as modelling, we shall introduce the term *modelling spaces* to denote, more formally, the kinds of workspaces set up by designers to carry out design tasks. Two kinds of modelling spaces can be identified from the scenario:

(I) Individual Modelling Spaces (IMSs) — the kind of workspaces where participants create and evolve to model design expressions targeted at a particular design aspect or domain of a design project. An individual's IMS may be physically

and/or logically separate from other individuals'.

(II) Group Modelling Space (GMS) — the kind of workspaces where members of a design team create and evolve to model the integration of design parts, as contributed by the individuals, into larger design wholes. A GMS is initially a public visual space for displaying individually made design expressions; a GMS may be developed to accommodate new elements and functionality emerging from direct or indirect communication among participants. The emerging elements and functions are essential to the realisation of design integration as intended by the group members

3.2 Modelling Acts

As shown in the scenario above, it can be said that designers perform actions of various kinds to produce, change, or evaluate states of design works. We now look into these actions of designing more closely in the following terms:

(a) *Abstraction*—the acts of forming a design representation scheme with which a designer establishes correspondences between his or her modelling space and the aspects of the artefact yet to be constructed in the real world.

(b) *Generation*—the acts of producing specific (concrete) design expressions (i.e., drawings, design specifications, etc.). In short, generation is about the use of a representation scheme by an individual's *design intents*.[2]

(c) *Interpretation*—the acts of assigning, associating, or calibrating the values (or certain meanings) of design expressions generated. The act of interpretation often involves a designer's referring to design knowledge developed and accumulated in certain design domains (e.g., building standards, ergonomics, material strengths, etc.).

(d) *Modification*—the acts of making changes in (parts of) the representation schemes abstracted or the design expressions generated. The acts of modification normally have the objectives of extending the scope of a representation scheme by introducing new elements or operations, and of changing the properties and relations of design instances constructed.

To add another dimension into our view of design as modelling, we may term the above design actions as 'modelling acts'. The entries of modelling acts listed above are four among others; and there are no obvious causal relations assumed between the acts. For our purpose of developing a descriptive theory, the four modelling acts included here are considered sufficient for the time being.

Modelling Acts / Modelling Spaces	*Abstraction*	*Generation*	*Interpretation*	*Modification*
IMSs	*Individual Object Worlds (IOW)*	*Local Design Decisions (LDD)*	*Domain Design Agendas (DDA)*	*Changes in LDD or in IOW*
GMS	*Shared Integration Schemas (SIS)*	*Common Images (CI)*	*Common Design Metaphors (CMD)*	*Changes in CI or in SIS*

Figure 2: An action-space matrix generating eight generic states of individual and group design work.

3.3 An Action-Space Matrix

In formulating the notions of modelling spaces and modelling acts, for the reason of convenience, we have separated them into two camps. As read in the scenario, modelling acts always take place in group or individual modelling spaces. We now put the two formulations together with the aim of constructing an *action-space matrix* (Figure 2). The matrix is presented to generate and classify eight generic states of individual and group design work.

3.4 The Flow of Information

For every single situation type generated in the action-space matrix, we have given explanations accordingly. The explanation thus carried out seems to suggest some constituents of a structural view of the metaphorist approach. As has been shown in our sequence of explanation, we have tacitly implied a kind of dependence relation among the situation types classified; i.e., some situations may follow if and only if others occur in the first instance. To see the kind of dependency more clearly, we need to put all the situation types presently classified back into the metaphorist scenario given earlier. In so doing, a schematic map of the *flow of information* in the metaphorist approach to collaborative design is constructed in Figure 3.

3.5 Constraints on Collaboration

According to situation theory, when once we have some idea about the flow of information among abstract or concrete situations classified for an activity, we are in a better position to systematically spell out the *constraints* (or logic) which governs that activity. In the final part of our exposition,

[2]The notion of design intents in the acts of generating design expressions should be emphasized, because a representation scheme on its own cannot explain why different (specific) expressions result even if the individuals employ the same representation scheme.

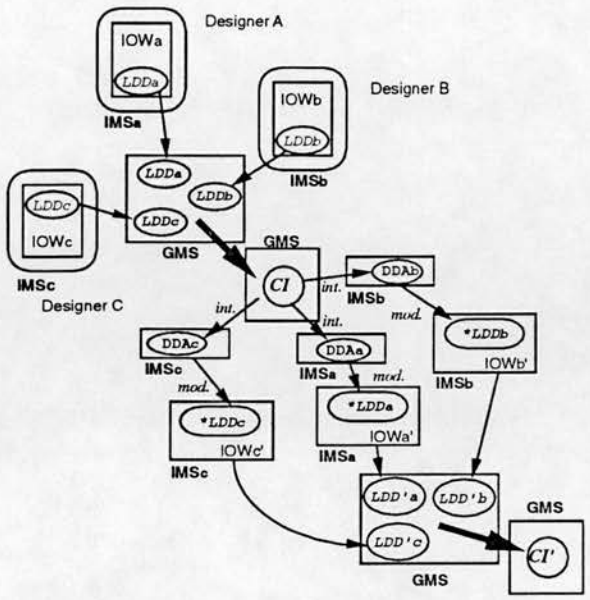


Figure 3: The flow of information among the situation types classified in the metaphorist approach to teamwork in building design.

a logic of the metaphorist approach to collaborative design is sought for. Firstly, in a slightly arbitrary way, the overall picture of the information flow gained above is individuated into four smaller sections. Upon each section, in the format of 'from to', an intermediate goal (task) of teamwork is focused; we then look into the general conditions for these tasks to be fulfilled by members of a design team.

(a) from *distributed local design decisions* ($\mathcal{LDD}_i$) to *a collective presentation of* $\mathcal{LDD}$ ($\Sigma\,\mathcal{LDD}_i$)—What is involved when members of a design team decide to jointly present their local design decisions to one another?

- A *call for participation* is sent out by a member (or members) to other members of a design team, which specifies when and where a design meeting will be held;
- In response to the call, team members do turn up for the meeting and bring along their latest design developments in various design domains;
- A *common visual space* is set up for displaying all participants' $\mathcal{LDD}$s such that potential relations or connections among parts of the $\mathcal{LDD}$s can be perceived and discussed among the participants.

The above three conditions point to the need of holding *design meetings*, in which design decisions made by different individuals in distributed sites are gathered together in a single workspace. From this, we may formulate our first constraint on collaborative design as follows:

$$Cons't\ 1 : ([\mathcal{LDD}_a, \mathcal{LDD}_b, \cdots] \rightsquigarrow \Sigma\mathcal{LDD}_i) \Longrightarrow Meeting$$

(b) from $\Sigma\mathcal{LDD}_i$ to *the generation of common images* ($\mathcal{CI}$)—Given a common visual space is available, what is involved in participants' arriving at common design images from their joint display of local design decisions?

When participants of a design meeting envisage that parts of their local design decisions can be interrelated in one way or another, common design images can be constructed in a shared workspace on the basis of gathered $\mathcal{LDD}$s. To realise the relations intended, the generation of $\mathcal{CI}$ may necessarily involve sets of *concepts for integration*, for example:

- concepts in terms of *new design elements* that can be deployed by participants when joining their local designs into a larger composition;
- concepts in terms of *spatial operations* that can be applied to geometrically transform parts of local designs prior to the final generation of integrated design images;
- concepts in terms of *projective operations* applicable to project certain kinds of over all design effect on the basis of multiple inputs of local design decisions.

As interpreted in our action-space matrix, *Shared Integration Schemas* may result from group members' joint abstraction of the means for realising the intended interrelations among individual works. Seen in the information flow framework, $\mathcal{SIS}$ has a functional role to play—elements of $\mathcal{SIS}$ constrain the flow of design information from gathered $\mathcal{LDD}$s to $\mathcal{CI}$.[3] For this reason, the second constraint on collaborative design may be expressed as follows:

$$Cons't\ 2 : (\Sigma\mathcal{LDD}_i \rightsquigarrow \mathcal{CI}) \Longrightarrow \mathcal{SIS}$$

(c) from $\mathcal{CI}$ to *distributed domain design agendas* ($\mathcal{DDA}_i$)—How do participants acquire their domain design agendas for further design developments in relation to the common images generated previously?

There are evidences from designers' retrospection showing that a state of $\mathcal{CI}$ can be mapped

[3] A further analysis into what is involved in participants' developing shared integration schemas is reported in [5].

onto (interpreted as) particular *templates* or *patterns* which serve members of a design team to achieve a sense of *wholeness*. The mapping (or interpretation) may be initially proposed by some individual(s), and then get recognised by the rest of the team members.[4] When such an interpretation of $\mathcal{CI}$ is commonly agreed by all participants, we say that a common design metaphor has emerged.

Here is a situation similar to the use of metaphors in ordinary conversations—the creative uses of some rhetorical devices for the purpose of communication. However, as mentioned before, the metaphors created and passed around by designers are more of the nature of imagery or graphics, hence the term 'design metaphors' is introduced.

Given a metaphorical framework emerging from group design processes, participants can always have individual interpretations of $\mathcal{CI}$ with a different purpose from inter-personal design communication.[5] An individual's interpretation of $\mathcal{CI}$ is mainly oriented toward a redefinition of existing relations, or an articulation of new relations, between individual contributions and emerging wholes.

This is like a participant's reflecting on (1) what his or her (new) local role is about, given an emerging design context rendered in $\mathcal{CDM}$; (2) what needs to be done to fulfil the role more exactly. Therefore, interpretation of this mode involves a domain-specific design perspective and knowledge that an individual is working with. A domain design agenda may arise from an individual's design enquiry of this kind. To different participants, the agendas acquired highlight design issues to be handled in refining or changing domain design decisions previously made.

Following the above account, we may point out that without the emergence of $\mathcal{CDM}$, it is less likely that each design party could draw up instant guiding agendas that are pertinent to continual domain design developments:

$$Cons't\ 3: (\mathcal{CI} \rightsquigarrow [\mathcal{DDA}_{\mathcal{A}}, \mathcal{DDA}_{\mathcal{B}}, \cdots]) \Longrightarrow \mathcal{CDM}$$

(d) from *individual or joint design changes in local design decisions* ($\Delta\Sigma\ \mathcal{LDD}_i$) to *design changes in common images* ($\Delta\ \mathcal{CI}$)—Assume that some design changes are made by some individuals (in accordance with their domain design agendas), how are the changes intended by individuals reflected in the changes of common images?

Given a newly acquired $\mathcal{DDA}$, a participant may proceed to develop his or her domain designs further, resulting in certain (intended) changes, regarding the existing $\mathcal{LDD}$, or more fundamentally, the current status of $\mathcal{IOW}$.

Due to participants' sharing a working protocol for design integration, changes intended in one design domain may have critical implications for the works pursued in other domains; that is, the 'repercussion' effect. Therefore, for an individual to be able to actually realise his or her intended changes, they have to be publicised to other participants for holding an 'exploratory integration'. By examining the consequent changes in the current state of $\mathcal{CI}$, participants can have their own (domain- oriented) judgements for supporting or rejecting the changes proposed. More specifically, we may think of the following examples of group interaction involved in making design changes:

- *backtracking*. The proposer has to drop the intended changes because some members cannot accept the outcome or the implications of the proposed changes from their own design perspectives;
- *competing*. The proposed design changes are not agreed by some other members but invite the members' design thinking, and they may subsequently produce alternative design changes that compete with the original ones;
- *coordinating*. Participants accept the result from an explorative integration and respond to the changes projected by making changes in relevant design domains to coordinate the proposed ones;
- *confirming*. The explored integration result judged satisfactory to all participants, and it does not demand further changes to be made in relevant design domains; participants simply send their confirmations to the proposer(s).

Viewed as the above, communications among team members are necessarily involved in the transition from changes in local design decisions to a new state of common images.[6] To better summarise what conditions information flow

[4] A deeper exposition of what constrains a person's ability to suggest such an interpretation and the abilities of others to recognise the proposed interpretation is beyond the scope of this paper.

[5] It is questionable if all participants can always exercise their individual interpretation of a state of $\mathcal{CI}$. This paper suggests that the association of a $\mathcal{CDM}$ with a state of $\mathcal{CI}$ is an essential attribute for allowing multiple interpretations of $\mathcal{CI}$.

[6] There lies a basic difference between the constraint of *Meeting* described earlier and the constraint *Consulting* identified here; the meeting constraint points to the joint abstraction of means for design integration, while the latter one points to the joint judge-

in this section, a *Consulting* constraint is expressed as follows:

$$Cons't\ 4 : (\Sigma\Delta\mathcal{LDD}_i \rightsquigarrow \Delta\mathcal{CI}) \Longrightarrow Consulting$$

4 Pointers to Developing Collaboration Support

Researchers of situation theory have demonstrated that a situation-theoretical modelling of human activities can lay a foundation for designing interactive information systems useful to the activities (see, e.g., [2, 4] among others). As a reserach agenda, the current study of the metaphorist pattern of creative human collaboration also suggests several supporting issues to be further investigated:

(1) *Support for joint presentation of local design decisions.* A *common visual space* is needed in which members of a design team can jointly present their latest domain design developments so that potential connections among locally developed works can be constantly envisaged by whoever participates in the meeting. More specifically, this suggests at least three mechanisms to be provided: the *networking* of remote workspaces, the *scheduling* of meeting, and the *filtering* of surface images from domain design expressions.

(2) *Support for joint abstraction of shared integration schemas.* An important requirement is that the construction of common images are always based on local design decisions as the source expressions. Therefore, a general *spatial* or *functional language* is needed so that participants can define new joint elements or operations by translating and combining domain concepts into shared integration schemas.

(3) *Support for joint interpretation of common images.* Currently, we don't have evidence showing that common design metaphors are represented in any explicit way but seemingly 'floating' among the meeting minds. However, a *distributed database* allowing participants to quickly retrieve *visual references* for current use can certainly enhance group interaction in recognizing the significance of a newly generated common image.

(4) *Support for consultation in making design changes.* Due to the operation of shared integration schemas, one member's design changes may cause further changes in other domains to be followed. A *housekeeping* mechanism can be developed to send alerts to group members when making design changes.

Acknowledgements This paper was written under the CSCD project at ITRI, University of Brighton. The author thanks the comments from Aart Bijl and John Lee at EdCAAD, University of Edinburgh. Thanks also go to the encouragement and support from the CSCD Group and Donia Scott at ITRI.

References

[1] J. Barwise. *The Situation in Logic.* CSLI Lecture Notes No. 17, Center for the Study of Language and Information, Stanford, 1989.

[2] J. Barwise and J. Etchemendy. Visual Information and Valid Reasoning. CSLI Report, Stanford, USA, 1990.

[3] J. Barwise and J. Perry. *Situations and Attitudes.* The MIT Press, 1983.

[4] K. Devlin. Situation theory and the design of interactive information systems. Technical Report CSLI-92-171, Center for the Study of Language and Information, July 1992.

[5] C. Peng. Cooperative Tasks of Modelling Spatial Requirements: An Algebraic Specification Analysis. Working paper, EdCAAD, Department of Architecture, University of Edinburgh, August 1992.

[6] C. Peng. Participatory architectural modeling: Common images and distributed design developments. In *Proceedings of the Participatory Design Conference 1992 (PDC'92), Kresge Auditorium, Massachusetts Institute of Technology, Cambridge MA US, November 6-7 1992*, pages 171–180. Computer Professionals for Social Responsibility (CPSR), 1992.

[7] C. Peng. Exploring communication in collaborative design: cooperative architectural modelling. *Design Studies*, 1993. (To appear).

[8] M. Speidel. *Team Zoo.* London: Thames and Hudson Ltd, 1991. Translated from the German by Michael Robinson. Translated from the Japanese by Manfred Speidel.

ments of the results of design integration.

Supporting Heterogeneous Distributed Substantiation of Common Generic Structures in Collaborative Design

Chengzhi PENG*
Information Technology Research Institute,
University of Brighton, Lewes Road,
Brighton, East Sussex, BN2 4AT, U.K.

Abstract

This paper presents an analysis of collaborative design activity, explaining how design participants collaborate to achieve integrated design on the basis of sharing and substantiating common generic structures with domain design developments. Referring to a previous observation of teamwork in architectural modelling, an overview of the *structuralist* approach to collaborative design is firstly introduced. The structuralist scenario is then classified into the aspects of model construction, model-constructing constraints, and modelling acts. By examining the properties of different types of design representations and the systematic relations among them, the *constraints on collaboration* are identified; a logic of collaborative design is found in the necessity of maintaining a dual correspondence between the evolution of *common generic structures* and the development of *domain design solutions* distributed over several sites. Following the constraints derived, a discussion of the basic requirements for computer-based tools to support collaborative design is given. The paper concludes with how the current work can be related to the research carried out in other areas.

Keywords: collaborative design, common generic structures, heterogeneity, distributed substantiation, computer support requirements.

1 Introduction

Research on designing computer-based tools to support group design activity has recently become active, attracting the attention of researchers working in various fields. Unlike conventional computer-aided design tools, the new tools are expected to support *collaboration* among designers participating in project work. This paper attempts to establish a conceptual framework for

*Formerly, EdCAAD, Department of Architecture, University of Edinburgh, 20 Chambers Street, Edinburgh EH1 1JZ, U.K.

the development of a computer-based modelling environment that can facilitate collaborative design activity.

Initial solutions to the design of communicating and computing tools that can be supportive for people involved in collaborative design can be seen in the field of Computer-Supported Cooperative Work (CSCW). In a recent bibliographical survey of the research in CSCW, Greenberg [Gre91] has introduced the key phrase "shared workspace". It is noticeable that a large portion of the experimentation on shared workspaces has to do with building prototypes of "shared drawing space". These group drawing tools were based on earlier observational studies of working group graphics and shared drawing space activities (see, for instance, [Lak83] [Bly88] [TL88]).

However, it is evident that, to different CSCW research groups and system developers, 'design' has different meanings. As revealed in the published prototypes, design activities have been described and analyzed according to various research interests and perspectives. Due to the differences in studying group work in design, there appears diversity in understanding of what constitutes a workspace for collaborative design; and the different understandings of 'how drawings and drawing activity are related to design' have resulted in varied system solutions to 'what is to be facilitated by a shared drawing tool'.

Our current enquiry into computer-supported collaborative design has an emphasis on *design thinking* in a teamwork context. For this purpose, it is considered that a move into the study of *design modelling activity* can open up a more appropriate research vantage. Seen from design modelling, a designer not only performs design actions but also considers *design representation* which is the designer's concerning with the ways of generating and modifying design results. We believe that an understanding of teamwork in design modelling can contribute to a unified framework for eliciting and organising both representational and communication requirements in collaborative design. In particular, we are interested in giving a structural account of the collaborative design activity that has the following features:

- *Integration* — The teamwork has a general goal of achieving single integrated designs that satisfy all participating design views;
- *Distribution* — The teamwork is participated by multiple individuals who hold different design perspectives and work in heterogeneous design worlds;
- *Evolution* — The teamwork is situated in the interplay between *integration* and *distribution* (i.e., no fixed routes of integration or of distribution are set in advance).

In an earlier case study of participatory architectural modelling, reported in [Pen92c] [Pen92b], a distinct teamwork approach to design modelling is characterised as "structuralist"[1]. Our

[1] "Structuralist" is the term coined by the author to characterise the observed pattern of teamwork in archi-

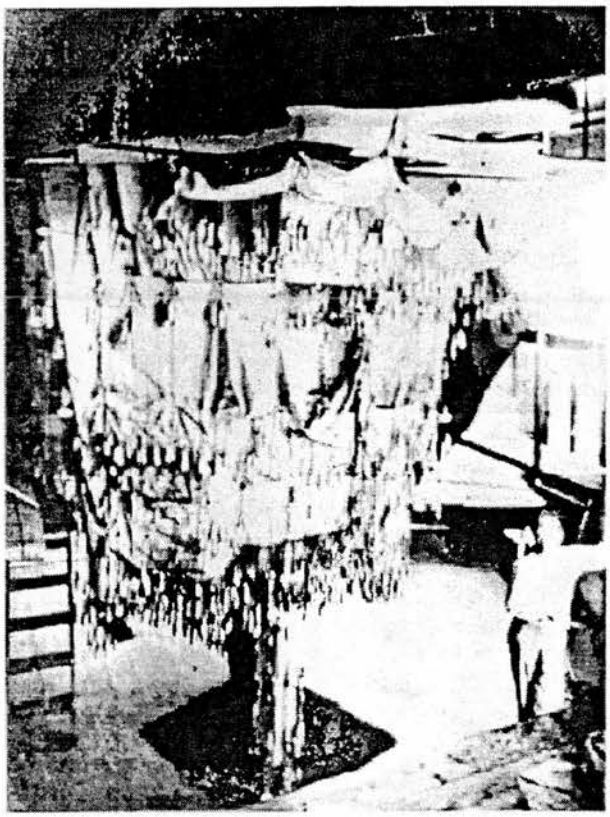


Figure 1: The construction and sharing of the funicular structure in the Colonia Güell church design project.

abstraction of the structuralist approach is mainly based on a study of the funicular model constructed in the Colonia Güell church design project recorded in [Mar79, p.335] and [CN83, pp.31-35] (See Figure 1). This approach presents an interesting feature of group design: some *common generic structures* are created collectively but viewed and substantiated differently by members of a design team.

The remainder of the paper is organised as follows. An overview of the structuralist approach is introduced in the next section. Given the structuralist scenario, Section 3 presents a classificatory scheme, individuating the basic constituents of the collaborative design activity. Within the framework, the constraints on collaboration are derived in Section 4. Following the constraints presentation, an informal specification of the requirements for computer support is presented in Section 5. The paper concludes with a discussion of related research and a direction to further work.

tectural design. As it will be shown later, the term is purely used to indicate the construction of "structural objects" in group modelling processes. It is not intended to relate to other notions of "Structuralism" developed by, e.g., Levi Strauss, Derrida or others.

2 The Structuralist Approach: An Abstract

By referring to our previous observation of the funicular modelling case, the structuralist approach to teamwork modelling can be described concisely as follows:

> At the inception of a design project, members of a design team work jointly in constructing a single spatial framework or skeleton as a *Common Generic Structure* in a *Group Modelling Space*. When applying projective devices onto a state of a common structure, *Derivative Structures* can be produced and then imported to *Individual Modelling Spaces* that are set up and used by different participating designers.
>
> By taking derivative structures as design referents, participants carry out separate strands of *Domain Design Developments* by substantiating (parts of) the generic structure into specific design expressions in individual modelling spaces. In the course of elaborating design developments, however, any participants may work up to the need to change parts of the derivative structures in use; to fulfill the intended changes, the individuals manipulate and modify parts of the common structure. The changes thus proposed by one individual can subsequently cause further changes to be made in the derivative structures used by other participants.

3 A Classification of the Structuralist Scenario

The special terms introduced in the above abstraction denote some of the primary concepts of the structuralist approach to collaborative design. In this approach, team members are enabled to coordinate domain design developments through the sharing of the generic structures constructed by themselves. The concepts arose here are important in soliciting the requirements for representing and communicating design intents in collaborative design. To analyse the scenario further, a more elaborate account of the concepts abstracted above is given below.

3.1 Common Generic Structures

Common Generic Structures ($\mathcal{CGS}$) are 2-D or 3-D generic objects, representing kinds of spatial frameworks or skeletons that are constructed and used by all participants working in different aspects of a design project. When created and evolved in a group modelling space (see below), parts of a common structure can be manipulated directly by participants working in different design domains. There are the following general properties of $\mathcal{CGS}$:

- *Deformability.* $\mathcal{CGS}$ are made as instances of model constructs that are connected in a field of physical forces or formal constraints. Being constructed and shared by all participants, $\mathcal{CGS}$ are meaningful and useful in revealing certain spatial forms or geometrical shapes.

Changes in forces/constraints applied or in values of constructs may *deform* $\mathcal{CGS}$ into different states. The deformability of $\mathcal{CGS}$ entails that the construction of the generic objects is based on certain *physical* or *formal models* which behave in certain systems of constraint satisfaction or equilibrium of forces.

- *Multiple-viewpoint.* Parts of $\mathcal{CGS}$ can be manipulated by participants from multiple points of view for different design reasons. The multiplicity is firstly achieved by participants' introducing types of model constructs that correspond to various "perspectives" of design modelling (e.g. site, structure, enclosure, opening etc, in building design). Secondly, there are multiple ways allowed to assemble or detach model constructs while modifying parts of $\mathcal{CGS}$. This multiplicity lies in a range of *connecting devices* that can be used to associate various types of model constructs introduced in the first place.

- *Derivability.* A state of $\mathcal{CGS}$ can be applied with *projective devices* as intended by any individual designers. Valid projections of $\mathcal{CGS}$ can generate instances of derivative structures (see below) that can be further transported to individual workspaces for domain uses. The derivability of $\mathcal{CGS}$ can allow participants to establish *referencing* relations between individual design developments and the shared generic structures.

3.2 Group Modelling Space

The term *Group Modelling Space* (GMS) refers to a modelling space[2] created by designers participating in project work. One of the key functions of a GMS is its use by all participants in modelling $\mathcal{CGS}$. The basic constitution of a GMS may include the following components:

- *Model constructs* — a collection of elementary objects that can be instantiated[3] to represent what participants think of corresponding elements in the the real world.

- *Form-giving forces* or *constraints* — fields of physical forces or formal constraints that participants choose to shape or deform parts of $\mathcal{CGS}$. In designing buildings, examples of form- giving physical forces are gravity, acoustics, light etc, and certain spatial/shape grammars or schemas can act as systems of formal constraints. Given a constraining field in a GMS, all participants are concerned with if a state of $\mathcal{CGS}$, as manifested in a configuration of model constructs, is *equilibrium* or *satisfactory* to the forces or constraints applied.

- *Manipulative operations* — operations that enable participants to displace, transpose, or aggregate etc. instances of various types of model constructs such that common structures

[2] In its simplest constitution, we may think of a modelling space as a collection of model constructs and modelling operations that can be used to generate and modify design expressions.

[3] By "instantiation", we mean the assigning of particular values (e.g. values of length, weight, spatial position etc.) to types of objects.

can be created and evolved.

- *Projective operations* — operations that allow participants to perform certain spatial actions, such as sectioning, projecting, tracing, truncating, developing etc., so that "secondary" structures can be deduced.

3.3 Derivative Structures

Derivative Structures ($\mathcal{DS}$) are 2-D or 3-D pictorial objects representing kinds of spatial frames or skeletons. Images of $\mathcal{DS}$ are produced by participants' applying projective devices onto a state of $\mathcal{CGS}$. Once imported into individual workspaces, instances of $\mathcal{DS}$ can serve the individuals as referents in modelling domain design developments.

In the course of developing domain designs, there may be a need to manipulate the underlying referents. The manipulation may be direct or indirect. In the case of indirect manipulation of $\mathcal{DS}$, the imported referents are "frozen" images, and they cannot be manipulated in parts but only as a whole. To effect changes in the referents, designers make changes in $\mathcal{CGS}$ and then re-derive $\mathcal{DS}$ containing the intended changes.

In the case of direct manipulation of $\mathcal{DS}$, participants may have $\mathcal{DS}$ *re-interpreted* so that domain-specific transformation can be defined and operated with. However, in maintaining consistency in both cases, changes in $\mathcal{DS}$ should trigger corresponding changes to be reflected in $\mathcal{CGS}$.

3.4 Individual Modelling Spaces

Individual Modelling Spaces (IMSs) are modelling spaces created and used by individual participants for the development of domain design solutions. In general, designers set up IMSs to include the following components in dealing with individual design tasks:

- *Individual design world* — a designer's design resource, consisting of (domain-specific) notations and tools for coding, visualising, manipulating and evaluating design expressions.
- *Derivative structures base* — an information space for storing, sorting, and displaying the images derived from $\mathcal{CGS}$.
- *Communicating tools* — communication channels for receiving and issuing confirmations or disagreements of making changes in $\mathcal{CGS}$, as manifested in imported states of $\mathcal{DS}$, among team members.[4]

[4]It is supposed that individuals working in different design aspects may be located remotely from each other, and the "individuality" of the communicating tools lies in the "addresses" of the participating IMSs such that communications can be directed effectively across distributed work sites.

3.5 Domain Design Developments

As described, the structuralist approach to collaborative design starts with the the construction of shared generic structures. However, final design products often go well beyond the design of general structural skeletons. An equally important part of collaboration is concerned with how the generic structures can be substantiated with more specific substances or properties. In this respect, domain design developments are the specialisation processes that are normally carried out by multiple parties with different design expertises. Typically, there are the following modelling acts involved in developing domain design solutions:

- *Constructing domain expressions* — In their own ways, individual designers are contributing different aspects of design solutions that can be attributed to particular parts or layers of the common structures. An important common factor is that all domain designs are developed on the basis of $\mathcal{DS}$. That is, designers construct Domain Design Expressions ($\mathcal{DDE}$) by taking $\mathcal{DS}$ of interest as underlying design referents.[5]

- *Reviewing design consequences* — Since each $\mathcal{DDE}$ is developed in relation to what $\mathcal{DS}$ is underlaid, and any instance of $\mathcal{DS}$ is a projection of $\mathcal{CGS}$, the resultant $\mathcal{DDE}$, as viewed and judged by its author from a particular design perspective, is the *consequence* of $\mathcal{CGS}$.

- *Evolving shared* $\mathcal{CGS}$ — When an individual develops domain designs to a certain extent, on seeing the resultant $\mathcal{DDE}$, he or she may conclude that the underlying design referents are not satisfactory. To explore other possibilities, the individual searches for modifications in $\mathcal{DS}$. These intended changes will further affect the state of $\mathcal{CGS}$.[6] In this respect, participants' developing domain designs in distributed IMSs may actually contribute to the evolution of $\mathcal{CGS}$.

3.6 A Classificatory Scheme

In giving a more elaborate account for the structuralist concepts of collaborative design, we have started with an explanation of, mainly, what artifactual aspects are involved in the teamwork activity. However, as we may find, some notions emerged from our discussion need a better classification. That is, notions like *design constraints*, and *modelling actions* are in fact independent of the existence of the representational artifacts. A larger classifactory scheme for organizing these notions is needed.

Figure 2 thus presents a classificatory scheme that individuates the complex group design activity into a collection of simpler components. Seen in this scheme, design expressions are

[5] We may better characterise this individual modelling act as "constructing domain design expressions by *referencing* to parts of shared $\mathcal{CGS}$."

[6] In Section 4.2, we shall give more explanations of why this is always the case.

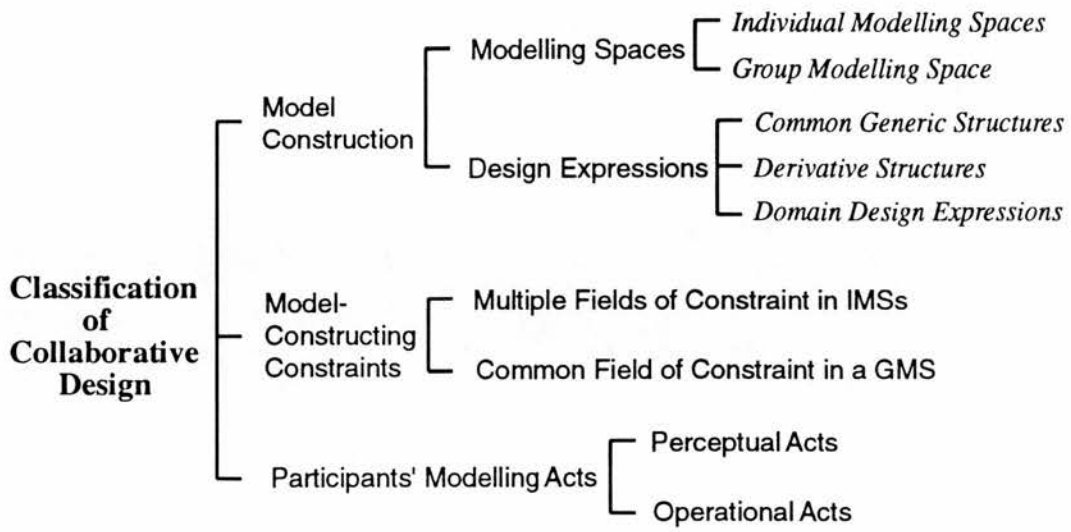


Figure 2: A classificatory scheme of collaborative design.

constructed and modified (by participants' modelling acts) in modelling spaces. When introduced into modelling spaces (group as well as individual ones), model-constructing constraints (e.g. gravitational forces, systems of spatial grammars etc.), can be applied to *deform* design expressions until an equilibrium (or satisfied) state is reached.

Members of a design team may perform two kinds of modelling acts. By *perceptual* acts, we mean that an individual's acts of constructing or manipulating parts of expressions is motivated (or caused) by his or her *perceiving* (or seeing and understanding) states of expressions. An individual can also perform *operational* acts to derive referential information from $\mathcal{CGS}$, or, more fundamentally, to update the constitution of modelling spaces.

Figure 3 illustrates how the components classified can form an abstract platform of collaborative design. The diagram summarises several features of teamwork in design that we have emphasised:

- *Heterogeneity in IMSs* — In modelling domain design solutions, participants employ individual modelling spaces, often equipped with domain-specific systems of design constraints. To serve individual purposes, the IMSs in use can be highly heterogeneous to each other.

- *Distributedness* — Each participant's modelling domain design solutions can be geographically and logically separate from the rest of team members'.

- *Structure sharing* — All participants share the states of $\mathcal{CGS}$ modelled in a GMS. The sharing of common structure is manifested in participants' capabilities of (a) accessing

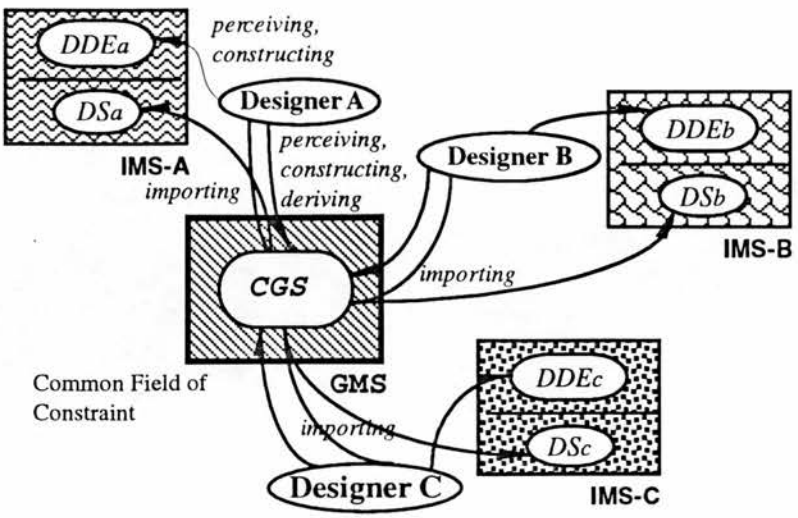


Figure 3: An abstract platform for collaborative design. (The number of participating designers is not definitive.)

to any parts of the structure, (b) changing the state of $\mathcal{CGS}$, and (c) extracting partial design structures for whatever design purposes.

The descriptions given in this section can be considered as an articulation of the 'infrastructural' aspects of teamwork in design. Given the rather static components arrived at, it's our next task to spell out the 'logical' aspects of collaborative design. In searching for the logic of collaboration in the next section, we aim to get a clearer picture of the more dynamic aspects of group design activity.

4 Constraints on Collaboration

In developing a semantic theory of natural language and information, Barwise expresses the following view [Bar89, p.52]:

> "When we search for the "logic" of some activity, what we are after is the collection of constraints $S \Rightarrow S'$ [7] that govern the activity. For example, the logic of perception consists of the set of constraints that govern perception."

Following the situation-theoretical stand-point, it seems plausible for us to think of the logic of collaborative design activity as a set of constraints[8] which governs collaboration. In the

[7] In Situation Theory [BP83], this is read as actual situations of type S which involve there being actual situations of type S'.

[8] Note that the notion of 'constraints' here has a different nature from that of design constraints described in Section 3.6. Model-constructing constraints are targeting at design problems framed in workspaces. Constraints

search for the constraints on collaboration, we find that a further examination of the properties and the systematic relations among different types of representations (i.e. $\mathcal{CGS}$, $\mathcal{DS}$, and $\mathcal{DEE}$) can tell us more about what is actually involved in collaboration.

4.1 Shareability of $\mathcal{CGS}$

The property of being "shareable" of common generic structure is a critical indicator of the continuing of teamwork. The shareability of $\mathcal{CGS}$ indicates the status of common understanding and judgement achieved and maintained by team members. In certain circumstances, $\mathcal{CGS}$ may become not shareable, hence teamwork cannot continue, due to the following reasons:

- Deformability — $\mathcal{CGS}$ may not be sustainable in a GMS because an equilibrium state of the structure under modelling cannot be reached;
- Multiple-viewpoint — $\mathcal{CGS}$ may not be accessible to some participants in the course of modelling because of the lack of certain types of model constructs or connectors;
- Genericity — $\mathcal{CGS}$ may not be usable to some members because of its derivative structures are not generic on a right level to serve the purposes of domain-related substantiations.

4.2 Consistency among $\mathcal{CGS}$, $\mathcal{DS}$, and $\mathcal{DDE}$

In our classification scheme, we have identified three different types of design expressions that are constructed by participants in different modelling spaces and serve for various purposes. However, in the course of constructing expressions, there exist certain "operational" relations such that the consistency of design information contained in different expression types needs to be maintained. More explicitly, consider the following operational relations:

1. $(\mathcal{CGS})R_d(\mathcal{DS})$ — Derivative structures are extracted expressions from a state of common generic structure. Therefore, $\mathcal{CGS}$ always stands in a relation, denoted as R_d, to $\mathcal{DS}$. The type of relation R_d can be characterised in terms of the derivative devices (methods) used and the spatio-temporal locations (relative to the $\mathcal{CGS}$ in a GMS) of applying the devices.
2. $(\mathcal{DS})R_f(\mathcal{DDE})$ — Domain design expressions are constructed by participants with reference to underlying derivative structures. Therefore, $\mathcal{DS}$ always stand in a relation, denoted as R_f, to $\mathcal{DDE}$. The type of relation R_f can be characterised in terms of '*referring to*', '*instantiating of*', or '*substantiating with*'.
3. $(\mathcal{CGS})R_d(\mathcal{DS})R_f(\mathcal{DDE})$ — Logically, due to the *information flow* between the types of expressions, a complex relation among $\mathcal{CGS}$, $\mathcal{DS}$, and $\mathcal{DDE}$ can be formed. Moreover, instances of the expression types can have different states in the courses of design modelling.

on collaboration are more to do with the problems of inter-personal communication.

As long as the relations R_d and R_f are in operation respectively, there can a problem of maintaining consistency. Consider further two types of design events:

$$\mathcal{CGS} \rightsquigarrow \mathcal{CGS}'\ (1)\ ; \quad \text{(some participant's act causing state changing in } \mathcal{CGS})$$
$$\mathcal{DS} \rightsquigarrow \mathcal{DS}'\ (2)\ ; \quad \text{(because a } R_d \text{ is in operation)}$$
$$\mathcal{DDE} \rightsquigarrow \mathcal{DDE}'(3)\ ; \quad \text{(because a } R_f \text{ is in operation)}$$

Or the event type can be the other way around

$$\mathcal{DDE} \rightsquigarrow \mathcal{DDE}'\ (1)\ ; \quad \text{(some participant's act causing state changing in } \mathcal{DDE})$$
$$\mathcal{DS} \rightsquigarrow \mathcal{DS}'\ (2)\ ; \quad \text{(because a } R_f \text{ is in operation)}$$
$$\mathcal{CGS} \rightsquigarrow \mathcal{CGS}'\ (3)\ ; \quad \text{(because a } R_d \text{ is in operation)}$$

Sharability of $\mathcal{CGS}$ and consistency maintenance among types of model expressions contribute to the main resources that give rise to constraints on collaboration in group modelling activity. On the basis of the above properties and relations examined, we spell out two constraints on collaboration in the following.

Suppose, at some design stage, a participant (say, Designer A) decides to make some changes in $\mathcal{DSs}_A$ (i.e. the set of derivative structures used by Designer A) to maintain (or validate) an intended domain design solution. Consequently, A's changing $\mathcal{DS}_A$ leads to a changing state of the $\mathcal{CGS}$ which is shared by other participants (say, Designers B and C). Owing to the deformability of the $\mathcal{CGS}$, the derivative structures, $\mathcal{DS}_B$ and $\mathcal{DS}_C$, used by B and C respectively may get changed in order to maintain the derivative relations introduced previously. This gives rise to at least two circumstances calling for communication among A, B and C:

1. [Constraint $Coor$: $Collaboration \Longrightarrow Coordination$]
   Coordination is involved if A's changing $\mathcal{DS}_A$ is seen in $\mathcal{CGS}$ and judged desirable by B and C, as they inspect the consequent states of their own derivative structures. Under this circumstance, B and C need to coordinate A's proposals by making further developments in their domain design solutions in respect of the changed $\mathcal{DS}_B$ and $\mathcal{DS}_C$.

2. [Constraint $Nego$: $Collaboration \Longrightarrow Negotiation$]
   Negotiation is involved if A's changing $\mathcal{DS}_A$ is judged undesirable by B and/or C, as they inspect the differences occurred in the changing states of $\mathcal{DS}_B$ and/or $\mathcal{DS}_C$. Under this circumstance, A needs to negotiate with B and/or C by either dropping completely the intended changes in $\mathcal{DS}_A$, which requires A to develop his or her domain design in a different direction, or requesting B's and/or C's suggestions of the extent to which the changes in $\mathcal{DS}_A$ are acceptable.[9]

[9]Note that A's receiving the suggestions made by B and/or C is same as how B and/or C may recognise A's intention expressed in changing $\mathcal{DS}_A$; the cognitive basis for A to do so is, again, the deformability of $\mathcal{CGS}$ and the relations between $\mathcal{CGS}$ and $\mathcal{DS}$.

5 Basic Requirements for Computer Support

Given the constraints on collaboration arrived at the above, a natural move forward is to specify requirements for computer supports, aiming at an ultimate design of a computer-based modelling environment that can support collaborative design activity. We may rightly ask "What do these constraints tell us about how collaborative design, as described in this paper, can be *computer-supported*?". However, we do not intend to present a formal and complete requirement specification at present. Instead, we now discuss some of the requirements, as prompted by our current analysis, for identifying areas of prospective computer support which have not been fully addressed or interconnected with other related research work.

(Issue 1) Support for the Construction of a GMS

As clearly revealed in the structuralist scenario, collaborative design begins with the construction of a common modelling space. Given the initial demand, a computer-based design environment may have to provide, in the first instance, representation supports in the users' construction of a GMS.

[1.1] *Representation of multiple viewpoints.* — More specifically, this requirement can be subdivided into the representation of two kinds of model objects:

[1.1.a] *Types of model constructs* — Participants working in various aspects of a project need to introduce types of model constructs that he or she considers pertinent representations of design elements. Different viewpoints in a GMS may be better represented by various types of *model constructs.* Instances of model constructs can interact with form-giving forces or constraints applied in a GMS and exhibit certain behaviours of deformation.

[1.1.b] *Types of model connectors* — When types of model constructs are introduced by participants, model connectors, the devices to connect or disconnect instances of the constructs, are essential. Types of connectors are used by participants to define and effect ways of manipulating parts of the common structures for various reasons. Note that model connectors are *neutral* objects in a sense that they do not represent any specific design elements in the real world.

[1.2] *Representation of constraint system for shaping $\mathcal{CGS}$.* — Design participants are not expected to build up, computationally, a common constraint system by themselves for modelling $\mathcal{CGS}$, since this demands highly technical knowledge. It would be a task for system engineers to develop computational models that can interact with instances of model constructs and connectors introduced by participants. The design of constraint systems of a GMS can be of the following nature:

[1.2.a] *General constraint systems* supporting physical (or, more broadly, environmental) laws such as gravity, thermal energy, or acoustics etc.

[1.2.b] *Specific constraint systems* supporting intentional laws such as particular systems of spatial or shape grammars.

[1.3] $\mathcal{CGS}$ *is pictorial and generic.* — Representation of $\mathcal{CGS}$ requires to be graphical and, at the same time, generic for the following two reasons:

[1.3.a] In serving *all* members of a design team as a common (global) representational medium, $\mathcal{CGS}$ is essentially *pictorial*, or, at least, *diagrammatical.* This implies that the construction of $\mathcal{CGS}$ has to be based on graphical objects so that all participants of different backgrounds can feel relatively easy to be familiar with.

[1.3.b] $\mathcal{CGS}$ is essentially generic in order to be enriched or refined to different levels of specificity. Therefore, its representation requires, perhaps, a higher order of genericity to support the following flow of information:

$$\mathcal{CGS} \overset{instantiation-of}{\longrightarrow} \mathcal{DS} \overset{substantiation-with}{\longrightarrow} \mathcal{DDE}$$

(Issue 2) Support for the Construction of IMSs

In teamwork, a participant's development of domain design solutions is not less important than that of common structures. To carry out more technical modelling tasks, participants need to work with *personal* workspaces which are not necessarily known and accessible to others. The problem is how to have a system capable of interacting with a user and generating an IMS which he or she thinks pertinent to the design tasks at hand. This requirement gives rise to the following sub-issues:

[2.1] *Representation of individual design worlds* — This includes, firstly, a set of personal design constructs for generating and manipulating domain design expressions, secondly, domain-oriented constraint systems employable in shaping domain design developments.

[2.2] *Support for the construction of* $\mathcal{DDE}$ *with reference to* $\mathcal{DS}$ — The spaces for constructing $\mathcal{DDE}$ is required to be *overlapped* or *juxtaposed* with the spaces for holding $\mathcal{DS}$ as design referents.

[2.3] *Support for the construction of* $\mathcal{DDE}$ *by substantiating* $\mathcal{DS}$ *with domain design elements (substances)* — This is a user's need for direct use of $\mathcal{DS}$ imported from a GMS. The type of construction process involves enriching or refining $\mathcal{DS}$ into $\mathcal{DDE}$ filled with more domain design details.

(Issue 3) Support for Coordination and Negotiation

The representations in a GMS and multiple IMSs discussed above are the infrastructural supports for the users' setting up group as well individual workspaces. Given the infrastructures outlined, we are in a position to spell out further system requirements of more

dynamic features. The third issue is concerned with system ability to support *coordination* and *negotiation* among design participants. In accordance with the constrains on collaboration explained in Section 4, the following communication requirements are expected to be fulfilled in the development of a collaborative modelling environment:

[3.1] *Detection of state change in $\mathcal{CGS}$* — It is clear to us that $\mathcal{CGS}$ is a dynamic object subject to participants' manipulations from different viewpoints. It is the evolving of a common structure that gives rise to the dynamism of teamwork. For a design environment to fit into the dynamic situation, it has to be concerned with the facts about state change in $\mathcal{CGS}$. But how do we define such a state change?

[3.1.a] A state of $\mathcal{CGS}$ is defined by a two or three dimensional deployment of instances of model constructs and connectors under the influence of a global constraint system activated in a GMS.

[3.1.b] A state change in $\mathcal{CGS}$ can therefore be defined as a change in (parts of) an existing deployment (or, a better word, configuration) resulting from a net effect of some participant's or participants' modelling actions together with the constraint influence.

A system's ability to keep track of state change in $\mathcal{CGS}$ lies in if the system can generate information about the configuration differences between two $\mathcal{CGS}$ states given at a time. This bit of information is essential for the system to trigger further communication mechanisms, such as the maintenance of R_d and messages delivering for users' maintaining R_f (see below). Seen in this requirement, a detection mechanism, so to speak, is needed.

[3.2] *Maintaining the relation R_d in $(\mathcal{CGS})R_d(\mathcal{DS})$* — In developing domain design solutions, participants need to extract derivative structures from a state of $\mathcal{CGS}$ as design resources or references. Since the state of $\mathcal{CGS}$ may keep changing, it is a useful support for participants if a system can inform the users timely the changing states of $\mathcal{DS}$ in use, arising from state change in $\mathcal{CGS}$. This requires a system to keep a record of the relation between $\mathcal{DS}$ and $\mathcal{CGS}$ and compute updated states of $\mathcal{DS}$ whenever $\mathcal{CGS}$ gets changed. Apart from the state of $\mathcal{CGS}$, two representations are necessarily involved in a system's maintaining the relation R_d:

[3.2.a] *Representation of derivative actions* — To derive a $\mathcal{DS}$, users require to perform certain *spatial operations*, such as *projecting, subdividing, or slicing* etc., upon $\mathcal{CGS}$. Taken as a bit of information, a *derivative action* thus consists of the performer and the spatial operation performed.

[3.2.b] *Representation of location of deriving* — The information about the time and position (relative to $\mathcal{CGS}$ modelled in a GMS) in which a derivative action takes place is also relevant in keeping a R_d.

[3.3] *Messages delivery for maintaining the relation R_f in $(\mathcal{DS})R_f(\mathcal{DDE})$* — Standing in a domain design perspective, a participant shall perceive his or her development of domain design solutions as design consequences in relation to a state of $\mathcal{CGS}$. By judging the development resulted, any participants may well be motivated to make changes in $\mathcal{DDE}$. This kind of design change activity gives rise to a second dynamism to the course of teamwork. As explained before, there exists the systematic relation, R_f, between $\mathcal{DS}$ and $\mathcal{DDE}$. Given a change in $\mathcal{DDE}$ desired by some individual, a R_f will not be sustainable if state changes in $\mathcal{DS}$, and hence in $\mathcal{CGS}$, are not reflected correspondingly.

A usable collaborative modelling environment should, therefore, not only allow for participants to freely make changes in $\mathcal{DDE}$ in their IMSs, but also assist the individuals in dealing with the problem of maintaining R_f. To support this communication need, two functionalities are considered necessary:

> [3.3.a] *Detection of state change in $\mathcal{DS}$* — A detection mechanism similar to that of detecting $\mathcal{CGS}$ state change is needed. But the detection functions need to be installed locally as IMSs may be distributed over a number of separate working sites.
>
> [3.3.b] *Sending the change message to GMS* — When a state change ($\mathcal{DS} \rightsquigarrow \mathcal{DS}'$) is computed, a message is sent to GMS for activating corresponding state change in $\mathcal{CGS}$.

When GMS receives and processes the message sent from IMSs, a change in $\mathcal{CGS}$ will be implemented by the system, resulting in $\mathcal{CGS}'$. Owing to the mechanism of maintaining R_d described in [3.2], further messages (containing the information about $\mathcal{DS} \rightsquigarrow \mathcal{DS}'$) sending from GMS to IMSs shall naturally follow so that other participants involved shall be informed. The detection and message delivery mechanisms described here seems to suggest a *local management agent* be set up in an IMS which is the sole information space for the agent to serve.

[3.4] *Communication channels for resolving conflicts manifested in $\mathcal{CGS} \rightsquigarrow \mathcal{CGS}'$* — If a coordination situation, as described previously on page 11, cannot be reached, negotiation among the individuals involved in the disagreement is needed to resolve the conflict. Since the situation is a highly non-deterministic one, a system is not expected to automatically detect the arising of a conflict and resolve it. In principle, this should be left to the participants to decide if coordinating or negotiating. In coordination, there is no need for participants to express individual judgements of the state of $\mathcal{CGS}$, and corresponding changes in $\mathcal{DDE}$ shall be carried out in IMSs separately.

More problematically, in negotiation, participants need to express disagreement to one another[10]. This demands a system to provide users with communication channels with which they can discuss, directly or indirectly, and resolve the differences in recognising the state of $\mathcal{CGS}$ until the sharability is re-established among members of a design team.

6 Related Research and Further Work

To investigate the possibility of computer-supported collaborative design, we have started from a study of the structuralist approach to teamwork in architectural modelling. By carrying out a natural analysis of the structuralist scenario, a classification scheme that explains the constitution of collaborative design activity is presented. Guided by an examination of the properties of the types of representation and the systematic relations among them, a logic of collaboration in teamwork is found. The constraints spell out what is involved when members of a design team co-work on the substantiation of a common generic structure with heterogeneous design developments in a distributed manner. Following the constraint presentation, we then give a discussion of the basic requirements for prospective computer supports.

For the purpose of drawing up a promising strategy for a further exploration, we have some readings from other researchers. In relation to our current enquiry, the following collection of research references are of a particular interest:

1. In their search for what makes research on Computer Supported Cooperative Work (CSCW) a unique research field, Schmidt and Bannon propose a general conceptual framework for CSCW (see [BS91, SB92]). In particular, they identify that the priority of computer support should be given to supporting a group of users for *articulation work* and the *construction of a common information space*. Our findings in supporting the structuralist approach to collaborative design appear in tune with the Schmidt-Bannon framework.

2. Based on analyses of organizational problem solving in scientific communities, Leigh Star derives the concept of *boundary objects* and suggests the concept would be an appropriate data structure for Distributed Artificial Intelligence (DAI) [Sta89]. Star identifies four types of boundary object which are considered as a major method of solving heterogeneous problems. Notably, the properties of boundary objects bear a close relation to those of our common generic structures [Sta89, p.46]:

   > "Boundary objects are objects that are both plastic enough to adapt to local needs and constraints of several parties employing them, yet robust enough to

[10] Again, to use the negotiation situation described on page 11, this is to say that B and/or C must find a way to let A know that A's intention in making the change in $\mathcal{CGS}$ is not acceptable.

maintain a common identity across sites. They are weakly structured in common use, and become strongly structured in individual-site use."

We suggest that the $\mathcal{CGS}$ in our case can be another candidate for a type of boundary object to be used in collaborative design but with a generic-specific structural adaptation instead of a weak-strong one. Though the general properties of boundary objects are researched, no computational representations of the objects have been proposed.

3. Research on computer graphics models, which can respond to in a natural way to applied forces or constraints, has shown us the technical possibilities of representing $\mathcal{CGS}$ graphically in computers. In particular, three research results worth noting: the theory of elasticity was employed by Terzopoulos *et al.* to construct elastically deformable models [TPBF87]; Witkin and others explored the representation of (geometrical) constraints as energy functions that behave like forces pulling and deforming parts of the model into place [WFB87]; three force-based constraint methods were explored by Platt and others to add several desirable properties into flexible models [PB88].

   It certainly remains to be seen how the kinds of graphics model achieved above can serve in a collaborative design context, satisfying the demands for being generic and manipulable for multiple design purposes. As for representing design constraints on a smaller scale, Gross and others developed constraint-based design environments as separate specialized design "Labs" (see [GEAF88] for detail). We see this work as a precedent experiment to the setting of private constraint systems in distributed IMSs.

4. System research and developments on computer-supported human communication in cooperative work have presented two distinct approaches: one is in favour of supporting informal interaction among users through the design of *shared virtual workspaces*; the other focuses on supporting formal interaction mediated by *communication protocols*. Research prototypes of shared drawing systems have demonstrated a range of technological means to re-create face-to-face communication where users are actually separated geographically (see, for example, [BM89, Lee90, TM91, LM91, IKG92] among many others, and [Pen92a] for a more detailed survey).

   Along with the second line, several computer-based coordinating protocols have been implemented. The building of these mechanisms is mainly based on a formal representation of either particular work procedures or special knowledge involved in the design tasks. In the domain of designing computer configuration for buildings, for example, the knowledge-based design environment *NETWORK* was implemented to test the idea of integrating the domain knowledge of configuring computer network and the communication between designers [FGL+92, RS92]. Bond and others developed a set of *rules of interaction*, arising from "an organisationally agreed sequence of commitment steps", to model the collaboration among specialists in aircraft design [Bon89, BR92].

It remains questionable, however, if the knowledge-based approach to the design of computer-based coordinating mechanisms can satisfy the needs of less stabilised group practice in design. The encapsulation of specific knowledge about artifacts or procedures can be problematic to collaborative design that demands unique solutions to every single project. Obviously, it is not very sensible to design a collaborative design environment centred on the funicular structure shown in the Colonia G'uell church project; as we know, there are always innovative building structures being developed.

The above overview of related research shows the complexity involved in developing a realistic collaborative design environment. It covers a very wide spectrum of conceptual and technical issues. To further our current work, we choose to focus on constructing a coordinating theory that is in tune with the representational and communication requirements elicited in this paper.

Acknowledgements This work was mainly carried out when the author was staying at Ed-CAAD, which was partially supported by the UK ORS Awards. The author wishes to thank the joint guidance from John R. Lee and Aart Bijl in developing the paper. Thanks also to the comments and supports from Lyn Pemberton, John Downie, Simon Shurville, and Donia Scott of the CSCD Group at the University of Brighton.

References

[Bar89] J. Barwise. *The Situation in Logic.* CSLI Lecture Notes No. 17, Center for the Study of Language and Information, Stanford, 1989.

[Bly88] S. L. Bly. A use of drawing surfaces in different collaborative settings. In I. Greif, editor, *Proceedings of the Conference on Computer- Supported Cooperative Work (CSCW'88)*, pages 250–256. ACM Press, 1988.

[BM89] S. L. Bly and S. L. Minneman. Commune: A shared drawing surface. Technical Report SSL-89-86, System Sciences Laboratory, Palo Alto Research Center, 1989.

[Bon89] A. H. Bond. The cooperation of experts in engineering design. In Les Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence Vol. 2 (Research Notes in Artificial Intelligence)*, pages 463–483. London: Pitman, 1989.

[BP83] J. Barwise and J. Perry. *Situations and Attitudes.* The MIT Press, 1983.

[BR92] A.H. Bond and R.J. Ricci. Cooperation in aircraft design. *Research in Engineering Design*, 4(4):115–130, 1992.

[BS91] L. J. Bannon and K. Schmidt. CSCW: Four characters in search of a context. In *Studies in Computer Supported Cooperative Work: Theory, Practice and Design*, pages 3–16. North-Holland: Elsevier Science Publishing Company, Inc, 1991.

[CN83] G. R. Collins and J. B. Nonell. *The Designs and Drawings of Antonio Gaudí*, pages 31–35. Princeton, N.J. Guildford: Princeton University Press, 1983.

[FGL+92] G. Fischer, J. Grudin, A. Lemke, R. McCall, J. Ostwald, B. Reeves, and F. Shipman. Supporting indirect, collaborative design with integrated knowledge-based design environments. *Human-Computer Interaction*, 1992. (To appear in Special Issue on Computer Supported Cooperative Work).

[GEAF88] M.D. Gross, S.M. Ervin, J.A. Anderson, and A. Fleisher. Constraints: Knowledge representation in design. *Design Studies*, 9(3):133–143, July 1988.

[Gre91] S. Greenberg. An annotated bibliography of Computer Supported Cooperative Work. *SIGCHI Bulletin*, 23(3):29–62, July 1991.

[IKG92] H. Ishii, M. Kobayashi, and J. Grudin. Integration of inter-personal space and shared workspace: ClearBoard design and experiments. In J. Turner and R. Kraut, editors, *Proceedings of CSCW 92*, pages 33–42. ACM Press, 1992.

[Lak83] F. Lakin. Measuring text-graphic activity. In *Proceedings of the GRAPHICS INTERFACE '83*, 1983. Edmonton, Alberta.

[Lee90] J. J. Lee. Xsketch: A multi-user sketching tool for X11. In *Proceedings of the Conference on Office Information Systems*, pages 169–173, 1990.

[LM91] I. M. Lu and M. Mantei. Idea management in a shared drawing tool. In L. Bannon, M. Robinson, and Schmit K., editors, *Proceedings of ECSCW'91*, pages 97–112. Kluwer Academic Publisher, 1991.

[Mar79] C. Martinell. *Gaudí: his Life, his Theories, his Work*, page 335. Barcelona Editorial Blume, 1979.

[PB88] J.C. Platt and A.H. Barr. Constraint methods for flexible models. *ACM Computer Graphics*, 22(4):279–288, 1988.

[Pen92a] C. Peng. A Survey of CSCW Designs in Shared Drawing Space. Working paper, EdCAAD, University of Edinburgh. vv In submission, June 1992.

[Pen92b] C. Peng. Exploring communication in collaborative design: A cooperative architectural modelling perspective. Working paper, EdCAAD, University of Edinburgh. To be published in the Journal of *Design Studies* on a special issue on collaborative design, 1993., September 1992.

[Pen92c] C. Peng. Participatory architectural modeling: Common images and distributed design developments. In *Proceedings of the Participatory Design Conference 1992*

(PDC'92), Kresge Auditorium, Massachusetts Institute of Technology, Cambridge MA US, November 6-7 1992, pages 171–180. Computer Professionals for Social Responsibility (CPSR), 1992.

[RS92] B. Reeves and F. Shipman. Supporting communication between designers with artifact-centered evolving information spaces. In J. Turner and R. Kraut, editors, *Proceedings of CSCW 92*, pages 394–401. ACM Press, 1992.

[SB92] K. Schmidt and L. Bannon. Taking cscw seriously: Supporting articulation work. *Computer Supported Cooperative Work*, 1(1-2):7–40, 1992.

[Sta89] S. L. Star. The structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving. In Les Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence Vol. 2 (Research Notes in Artificial Intelligence)*, pages 37–53. London: Pitman, 1989.

[TL88] J. C. Tang and L. J. Leifer. A framework for understanding the workspace activity of design teams. In *Proceedings of the Conference on Computer- Supported Cooperative Work (CSCW '88)*, pages 26–28. ACM Press, 1988.

[TM91] J. C. Tang and S. L. Minneman. Videowhiteboard: Video shadows to support remote collaboration. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 315–322. ACM Press, 1991.

[TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *ACM Computer Graphics*, 21(4):205–214, July 1987.

[WFB87] A. Witkin, K. Fleischer, and A. Barr. Energy constraints on parameterized models. *ACM Computer Graphics*, 21(4):225–232, July 1987.

Participatory Architectural Modeling: Common Images and Distributed Design Developments

Chengzhi Peng

Edinburgh Computer-Aided Architectural Design
Department of Architecture, University of Edinburgh
20 Chambers Street, Edinburgh EH1 1JZ, Scotland, U.K.
Tel: +44 31 6502339
Email: peng@caad.ed.ac.uk

Abstract

Drawing on cases of teamwork in architectural modeling, this paper reports a study in rich and informative approaches to participatory design. Two features of participation and coordination among designers are observed: (1) for technical necessities, members of a design team work in individual design worlds calling upon heterogeneous conceptual structures and instruments; and (2) for critical judgements, the emergence of final unity in design products as a whole is of common concern shared by all participants, which is dynamically related to the developments of design solutions in individual domains. By abstracting generic patterns of cooperative modeling from the cases discussed, several concepts of communication in participatory design are explored. It is found that a view of situating modeling acts in coupled modeling spaces can lead to a useful exposition of participatory design in terms of the interrelations between *common images* and *distributed design developments*. As guiding pointers to further research, the current study identifies two distinct generic patterns of communication in participatory design.

Keywords

architectural modeling, teamwork, modeling acts, modeling spaces, common images, heterogeneity, distributed design developments.

1 Introduction

In the field of design study, there are two ways of seeing participatory architectural design. One sees the participation among prospective users, professional designers, and clients etc, taking part in the planning, design, or even construction processes (see [Ers80, AHMC85, Kro88], among many other world-wide examples). The second focuses on participation by professional designers, for which various interpretative frameworks of how designers work with one another have been proposed. In this paper, I attempt to define participatory design from an architectural modeling perspective with the participation in the second sense mentioned.

In *PDC'92: Proceedings of the Participatory Design Conference*. M.J. Muller, S. Kuhn, and J.A. Meskill (Eds.). Cambridge MA US, 6-7 November 1992. Computer Professionals for Social Responsibility, P.O. Box 717, Palo Alto CA 94302-0717 US, cpsr@csli.stanford.edu.

However, in the absence of "user participation", the usefulness of this study remains justifiable by two points: (1) it demonstrates non-trivial examples of design perspective and undertakings of highly heterogeneous natures that often participate in the practice, and (2) it asks for an exposition of the relationships between the emergence of design products as a whole and the integration of partial solutions developed in distributed individual object worlds.[1] It can be argued that the issues of the nature confronted here may occur in most occasions of participatory design involving users; and it can be even more relevant if the design and use of computing tools is considered, as contemplated in [Gru91].

On interpreting participatory design of a narrower scope, one of the well-known metaphors is design as *game*. Lawson has reviewed several design games that were specially devised to model group dynamics in architectural and urban design [Law80]. Following the game paradigm, Habraken and Gross invented a computer program called *Concept design game* which can record players' interactive moves during sessions of control distribution and territorial organization [HG88]. Schon proposed a theory of *reflection-in-action*, drawing on one of his protocol analyses of a design dialogue between an architecture student and a studio master [Sch85. A similar studio-based study of participatory architectural design on a larger scale was carried out by Ward, in which seven subjects went through group processes and developed *archetypes* for a commercial complex project by gathering individually made cardboard models [War87].

Assuming that professional designers participating in a project are capable of communicating and acquiring user requirements, this paper adopts a somewhat different measurement toward an analysis of the communicative aspects of participatory design. Basically, it is considered that an adequate understanding of participatory design can be gained by studying, not simulated nor controlled but naturally developed, design expressions drawn from real cases of group practice. In particular, the largely graphical expressions are examined from a *modeling* point of view.

[1] The term and concept of "individual object worlds" is borrowed from Bucciarelli's recent ethnographic study of engineering design. For a more detailed description, see [Buc88].

That is, design is mainly taken as an activity of modeling complex objects. The drawings, diagrams or 3-dimensional models are the artifacts that designers produce to convey and coordinate individual design intentions and judgements.

As a general finding from analysing the artifacts, a long and heterogeneous participatory design process is said to involve two things: one is the emergence of *common images* that is shared among participants; the other is the developments of *domain design specifications* that are distributed over participants' individual modeling spaces. Having introduced common images first does not mean that what is common has to be developed in the first place; as shown in this study, it can be the other way around. By further inquiring into the interrelations between the modeling of the common and of the distributed, a general setting of participatory modeling is concluded. When situated in this generic setting, a number of modeling acts performed by participants are observed to demonstrate certain communicative properties.

The remainder of the paper is arranged as follows. In Section 2, a study of three examples of participatory architectural modeling, taken from real cases of group practice in design, is presented. Based on the case discussions, Section 3 gives an exposition of how common images are formed in relation to distributed developments of domain design specifications. The implications of two abstract patterns of communication identified in this paper for further research into computer-supported cooperative design is discussed in Section 4.

2 Examples of Participatory Architectural Modeling

Architectural modeling can take place in various dimensions, allowing for a variety of approaches to participation. What follows in this section is an introduction to this variety through three case studies of participatory architectural modeling. The first shows an example of one dimensional convergence of two conceptual design worlds participated in a fountain design project. The second shows an approach of overlaying two-dimensional diagrams constructed by at least three different design disciplines for re-engineering a large industrial building. The third case gives an exceptional illustration of a three-dimensional funicular model that was commonly constructed but used differently by a number of designers for a church design.

2.1 (Case 1) Between score and diagram

Design project: Seattle Center Fountain, Seattle, USA, 1962-1964.

Aspects & participants: waterscape design by two landscape architects (Lawrence Halprin, Curtis Schreier); and fountain engineering by a mechanical engineer (Daniel Yanow) [Hal69].

The scoring and diagraming spaces

- The Landscape Architects (LA) used a particular representation scheme called "score" for modeling fountain patterns and actions in a temporal frame (Fig. 1). A score has two dimensions: one for regulating multiple temporal sequences, represented in certain lengths of bars; the other for configuring spatial structures of different fountain stages (platforms), represented as point, square cross, rectangle etc. By manipulating the bars, a score reveals different compositions of active fountain stages against the inactive ones over a period of time.

- The Mechanical Engineer (ME) used "diagrams" to model mechanical components for piping, jetting, sprinkling design (Fig. 2). A pool piping grid was composed in a system of graphical symbols, corresponding to a set of design objects whose attributes were specified in words and numerals. In relation to the piping grid, a mechanical section was constructed to convey sectional information. Due to the correspondence set up between the mechanical components and the graphical symbols, the ME could virtually change the attributes and relations of particular design objects by manipulating parts of the diagrams.

A common space for projecting water effects

The graphical expressions in Fig. 3 shows a series of *squiggles* spreading over a formal *grid*. This evidence implies that a common modeling space shared by LA and ME was formed on the basis of combining the designs in the score and in the diagram, in which a sequence of water effects can be *projected*. Here we see an example of a set of *common images generated*, allowing for *interpretations* of the design consequences from various viewpoints. It is clear that the images of water effects can be interpreted both in LA's view -- the actions of fountain stages as scored over a time span, and in ME's view -- the fountain kinematics concerning the motions of pipes, jet heads, sprinklers as configured in the piping grid and mechanical section.

Interrelations between modeling spaces

Given the above evidence, two interrelations between scoring, diagraming, and projecting spaces are worth noting, which yields further accounts of what constitutes participation in developing the fountain design:

- Sequences of water effects at particular moments cannot be projected solely in LA's scoring space nor in ME's diagraming space; the possibility of projecting is conditioned by knowing what fountain stages are active at those moments and what mechanical devices are operative on those active stages plus how they shall behave -- a convergence between two individual modeling spaces whenever a projection is undertaken.

- Modeling actions taken in individual spaces change not only the state of the score or the diagram but also the state of the common image when projected; ME may take further actions upon his interpretation of the changing water effects propagated from LA's actions of changing score, and vice versa -- communication and coordination are called for to resolve disagreements or conflicts thus arising.

2.2 (Case 2) Participation through overlay diagraming

Design project: Cummins Research and Engineering Center, Indiana, USA, 1964-1968.

Aspects & participants: Structural Engineering (SE) (The Engineers Collaborative; Lighting Engineering (LE)

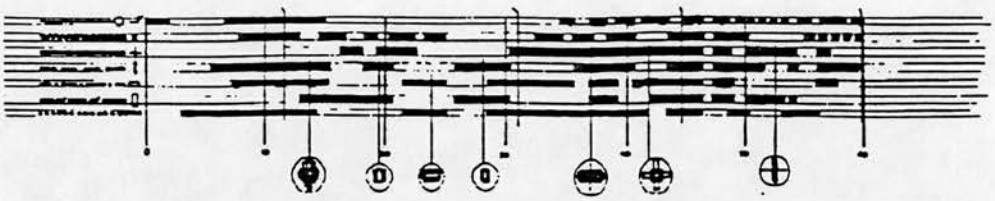


Fig. 1 The landscape designers' introducing and operating with *Score* in modeling fountain patterns and actions over a period of time. (Drawing taken from [Hal69], page 56)

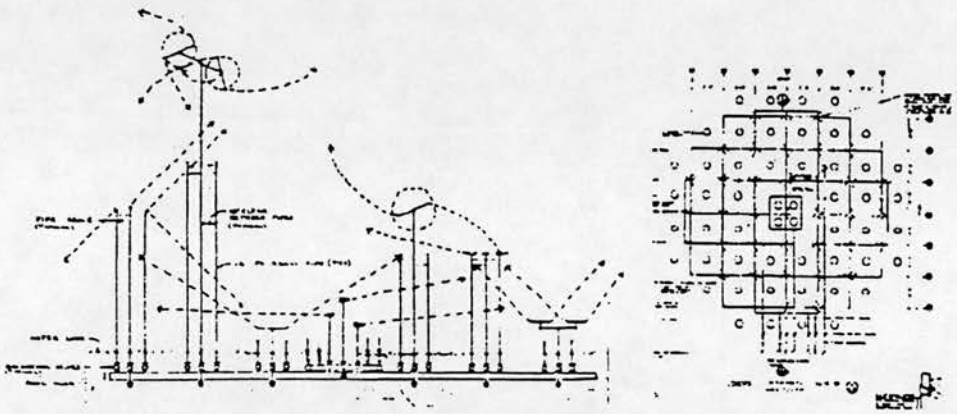


Fig. 2 The mechanical engineer's introducing and operating with *piping diagrams* in modeling the behaviors of the mechanical components. (Drawing taken from [Hal69], page 56)

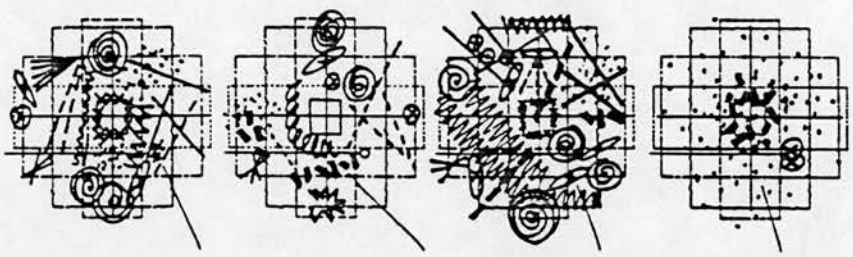


Fig. 3 The graphical indications of a shared fountain modeling formed by a combination of LA's scoring and ME's diagraming, which can project water effects, allowing for different interpretations. (Drawing taken from [Hal69], page 56)

(William Lam Associates); Mechanical Engineering (ME) (Cosentini Associates).

The main design issue is centered on how to "rearrange ductwork to the structure and to baffle the indirect light sources" [Lam77].

Distributed diagraming spaces

Each engineering discipline had its own object-based diagraming space.There were at least three domain-oriented diagraming spaces participating in the project: SE, ME, and LE (Fig. 4). Each diagraming space employed a special coding system to represent the modeled building components.

Evolving the environmental design through overlaying

According to Lam, the group processes evolved a "fishbone layout" which proved to be economical and satisfactory to all participants [Lam77] (see Fig. 5). Clearly, the emerging of the fishbone image is conditioned by the participants' continuously *overlaying* their individual diagrams.

Articulation of common images

Apart from the interrelations noted in Case 1, the current case shows that participants can further articulate a common image into various parts that play different roles or functions (the spinal cord or ribs of a fishbone, for example); differentiated portions of a common image are then distributed to serve individuals' developing domain solutions. Participation in design is therefore maintained by a *to and from* relation between the individual and the common which is in turn built up by the following processes:

- *overlay diagram construction:* A participant can construct diagrams on top of extracted common images

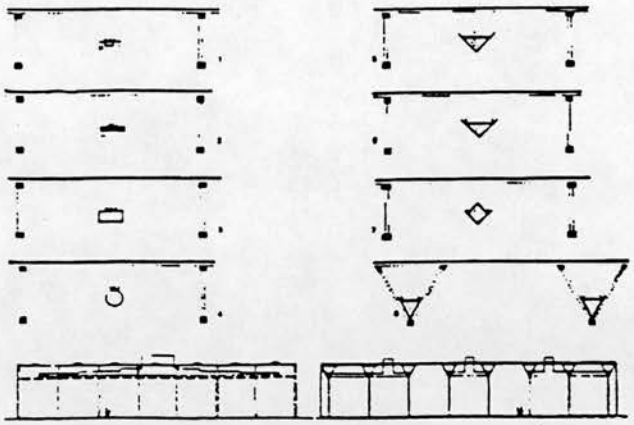


Fig. 4 Multiple diagraming spaces in different layers showing the participants' heterogeneous coding systems for modeling the aspects of the building design. (Drawings taken form [Lam77], p. 126)

which may contain parts of diagrams drawn by other designers working on different aspects.

- *overlay design checking:* Participatory design can be evaluated by checking overlaid consequences according to certain criteria such as detection of spatial clashes.
- *overlay design amendment:* A participant can modify parts of diagrams by referring to the diagrams underlaid in various ways (e.g. geometrically, economically, or aesthetically etc.); and one designer's amendments may cause related changes to be made by others.

2.3 (Case 3) Funicular modeling revisited

Design project: The Colonia Guell Church, Barcelona, Spain, 1889-1914.

Aspects and the participants: site planning and structural form by architects (Antonio Gaudi, Jose Canaleta); structural engineering by a civil engineer (Eduardo Goetz); ornamentation by a sculptor (Juan Bertran)

The funicular modeling space

An upside-down funicular model was constructed by the design participants at the inception of the project. According to Collins and Nonell [CN83], this large 3-dimensional model, which was shared and manipulated by all participants for different design tasks, had the following distinctive types of model components (see Fig. 6):

- *cords* hung in loops corresponding upside down to the placement and shapes of the piers and arches of the building's vault;
- several pieces of irregularly shaped (wooden) *boards* fixed onto the structure of the workshop, representing contour lines of the building site;
- *weights* made of pellets contained in small sacks (measured in the scale of 1/10,000), when attached to the hung cords, distorting the cords' catenary curves into funicular polygons;

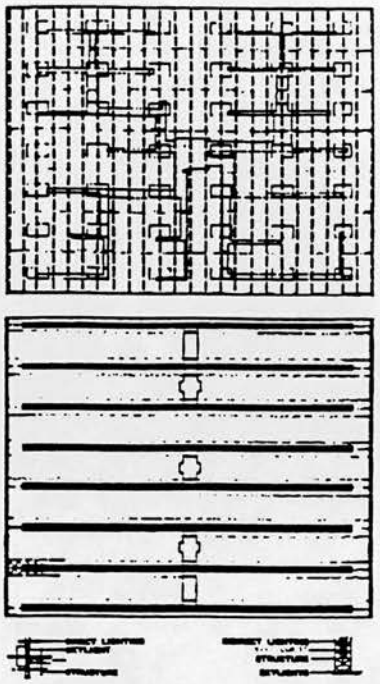


Fig. 5 The combined images of structural, mechanical, lighting design solutions get evolved through the participants' overlaying individual developments. (Drawing taken from [Lam77], p. 126)

- *fabric* draped onto the web of funicular polygons, representing the volumetric effects of the building exterior;
- a set of domain-neutral objects made of *jointers*, *hooks*, and *clippers*, which do not represent any particular components of the building design but function importantly in connecting the model objects and in manipulating parts of the funicular model.[2]

Funicular modeling and distributed drawing spaces

- The civil engineer's structural calculations: the distribution of loads in space and the thrusts of force lines were calculated by the engineer in a 2D vector space; to him, the funicular model was a 3D illustration of his 2D graphic static modeling.
- The architects' sketching out the exterior and interior spaces: photographs of the exterior and interior of the funicular model were taken and turned right-side up by the architects as the underlay information for modeling the locations, proportions, and shapes of openings (the fenestration of the building).
- The sculptor's sketching out the ornamentations: the sculptor was concerned with the design of sculptural objects such as the ornaments for the building's exterior and interior; like the architects, he took photographs of the funicular models for his own design interests and tried out design solutions by overlay sketching (Fig. 7).

[2] Jointers were used for attaching weights to cords; hooks for connecting the ends of cords to particular locations on the boards; clippers for clipping cords together at various heights (bifurcation).

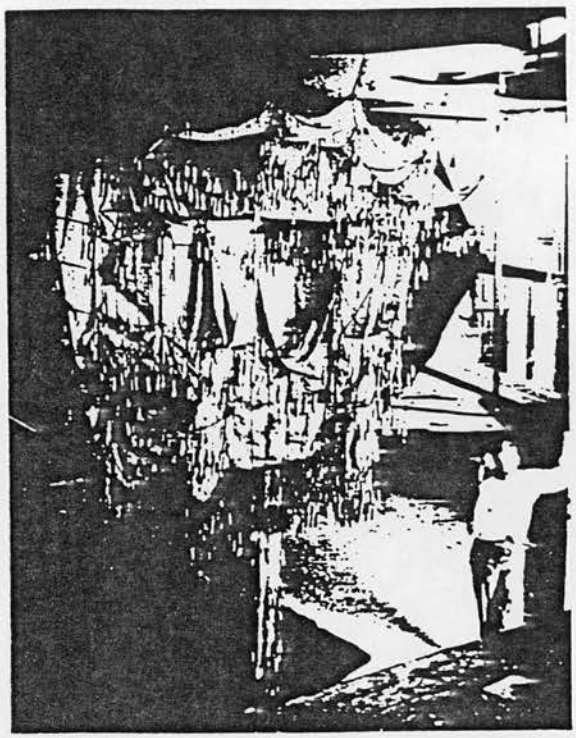


Fig.6 The funicular model constructed for the Colonia Guell church project as it hung in the workshed. (Picture taken from [CN83], Fig. 39)

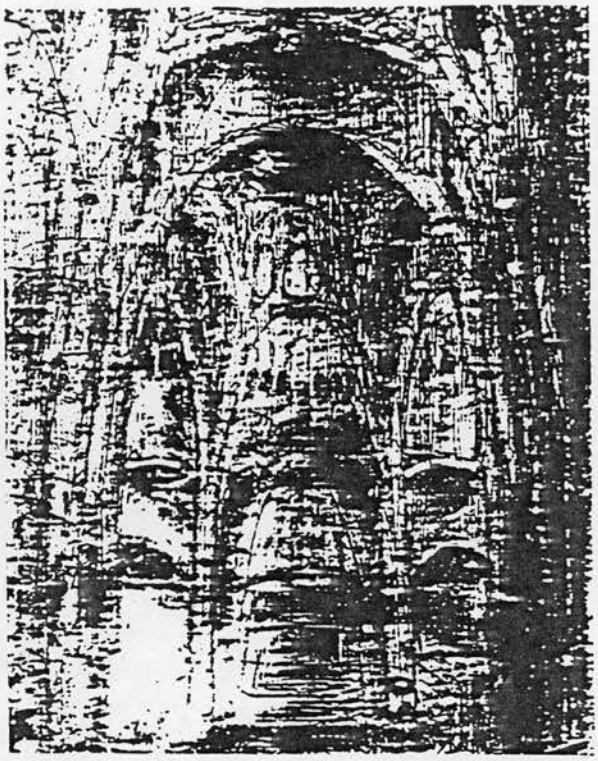


Fig. 7 A freehand sketch of the church interior design, tried out on top of a photograph of the funicular model inverted. (Drawing taken from [CN83], Pl. 57)

Group interaction in the funicular modeling space

Given the above observations, several accounts can be made for what makes the funicular modeling space a shared workspace for the design team, and how the shared model served as an evidence of interaction between the participants:

- First of all, the funicular modeling space was continuously developed and used by the design team for supporting long term participation; the participants collaborated on the modeling in delicate exploratory work lasting over 10 years [Mar79].
- The model served as a common image of a structural form shared by the participants, since the modeling space allowed them to manipulate the funicular model for reasons other than the strictly structural.[3]
- For any state of the model, the participants could have individual interpretations and derive design information from, perhaps, different measurements; and the information derived further served as the basis for the individuals to elaborate individual design models distributed over several work settings.
- The factor that the earth's gravity was one of the (direct) forces in shaping the model can explain how the group interaction could be coordinated by the shared modeling space. Through the action of G-force, the model constructed and manipulated always conforms to the physical law of *funicular structure* [Sch80]. Therefore, a modeling action taken by an individual, for whatever reason, can motivate or activate other team members' interpretations and actions in response to the changing state of the funicular model.

[3] For instance, for the purpose of site planning, cords can be shifted to different hooks or by moving the hooks around the board; for modifying fenestration design, cords can be bifurcated at various heights via sliding the clippers along the force lines; for changing structural form, loads can be redistributed in space by controlling the number of pellets in sacks or by displacing the sacks' jointers to different positions on cords.

3 Modeling Complex Objects by Participation: An Exposition

Alternative conceptual frameworks for describing the communicative aspects of group working have been proposed by researchers working in Computer-Supported Cooperative Work (see e.g. [TL88,Kuu91,SB91] among many others). The proposed frameworks are useful in two respects: first, they serve to describe, in more precise terms, the complex human phenomena, processes, or activities observed in practice, and produce coherent overviews; second, they indicate a range of requirements for guiding the developments of prototype computing or communication tools. In this section, by abstracting the previous case observations, a conceptual framework of modeling design objects by participation is proposed. Starting with the basic notions of modeling acts and modeling spaces, three communicative aspects of participatory design are explored. As a result of the exposition, two generic patterns of participation are found, which indicate several issues for further investigation.

Given the observations in the preceding section, design in general may be better characterized as an open-ended *modeling* process consisting of the following basic ele-

ments: (a) design constructs or concepts are (continuously) introduced and (re)structured, i.e. the forming of a design modeling space; and (b) shapes or forms of design artifacts are attributed, manipulated, and evaluated iteratively, i.e. the performing of modeling acts. Therefore, design as an activity can be conceptualized as the performing of modeling acts in a modeling space. In a long participatory process involving heterogeneous sources of design knowledge and actions, another distinction can be made among the modeling spaces that are formed/used by individuals and those by all participants: (c) multiple Individual Modeling Spaces (IMSs) are separated but logically and/or functionally connected with a Group Modeling Space (GMS).

By combining the above setting with the different types of artifacts identified, a space-action framework for characterizing participatory modeling can be set forth. The framework is constructed in three parts, presented in the following subsections.

3.1 Common images in a GMS

The fact that participants speak heterogeneous design languages does not prevent them from achieving design images that are commonly shared among them. Taking in various forms, common images can serve either as conceptual structures or as specific instances which allow for individual interpretations from different viewpoints. As observed, in modeling common images, expressions can be constructed in a group modeling space by the following approaches:

- A common image is constructed jointly by all participants, employing a shared construction method or system; serving as a shared conceptual structure, a common image allows for each participant's deriving and distributing its parts over individual modeling spaces.
- Common images are formed by participants' combining and integrating domain-specific design expressions with perhaps heterogeneous underlying structures; serving as the outcomes of participation and collaboration, common images are inspected and interpreted by the individuals for reflecting on design consequences from various viewpoints.

The state of a common image is subject to continuous changes that may well motivate intensive group communication in respect of how the image may be formed:

- Changes can be made directly in parts of a common image by any participant, if its existence originates from a group modeling space; and changes made by one individual in parts of a common image may have consequences somehow meaningful to other individuals' modeling spaces.
- Changes can be effected indirectly in parts of a common image, if it is formed on the basis of combined and integrated domain structures. The state of a common image gets updated by way of one or more participants' manipulating parts of domain expressions; a changing common image caused by one individual in an individual modeling space may thus motivate other participants' further actions in respect of the changing states of domain expressions.

3.2 Distributed design developments in IMSs

Design calls for participation mainly because its development requires a combination of design judgements and technical specializations that in reality one individual can hardly have. In parallel to the forming and evolving common images, aspects of a design project are often developed by designers trained with different design disciplines in a logically and/or geographically distributed manner. From a local perspective, design developments can be distributed according to two approaches:

- Participants employ individual working methods or object worlds in their own modeling spaces which are not necessarily known to each other; design expressions specific to certain modeling aspects are thus produced by perhaps markedly different individuals which then serve as the basis of joint construction of common images.
- Participants interpret the states of common images from different viewpoints; the structural images thus derived provide the basis for further domain-oriented design elaborations using modeling methods appropriate to the tasks.

Design changes targeted at local design developments can take varied accesses regarding how they may be developed, and the actions of making changes need to be interactive since they all have something to do with common images in group modeling spaces:

- Changing parts of domain design expressions constructed on top of derivative structures needs to be based on participants' manipulating corresponding parts of common images which may consequently change parts of underlying structures distributed in other design domains.
- Changes in local developments can be made directly in individual modeling spaces which may lead to changes in a common image that, in turn, motivate participants' taking modeling actions in related domains.

3.3 Interrelations between the common and the distributed

In the above, the artifactual aspects of participatory design including those of Common Images (*CI*) in a Group Modeling Space (GMS) and of Domain Design Expressions (*DDEs*) in multiple Individual Modeling Spaces (IMSs), are described in conceptual terms. Given these concepts, a natural question to ask is how these aspects are interrelated with each other. To draw the interrelations, two further concepts need to be introduced explicitly: the participants' performing a range of *modeling acts*, and the coupling of modeling spaces.

Performing modeling acts

A number of distinct actions involved in design modeling can be drawn from the case studies, for example:

- *Representing* -- involving, first, listing a collection of basic objects (constructs) which represents analogically or symbolically the corresponding elements of a design artifact; secondly, specifying how instances of the primitives are related in terms of what *operations* are applicable to the constructs.The act of representing leads to a *conceptual structure* of a modeling space.

Spaces Couping / Modeling Acts	*CI* / GMS → *DDEs* / IMSs	*DDEs* / IMSs → *CI* / GMS
Representing	- (L:) a shared conceptual structure made of group objects and operations - (R:) distributed derivatives of common images imported as underlying conceptual structures for domain uses	- (L:) individual conceptual structure made of heterogeneous sets of objects and operations - (R:) integrating individual conceptual structures into common sets of objects and operations
Mapping	- applying deductive, projective means on states of common images and acquiring derived structures for individual purposes	- translating parts of individual conceptual structures in one design domain into another for interpersonal/group purposes
Constructing	- (L:) changing states of common images yieled from applying group opertaions onto group objects - (R:) changing states of domain expressions yielded by applying some coding devices onto distributed derivative structures	- (L:) changing states of domain design expressions yielded in individual object worlds - (R:) changing states of common images yielded from combining/integrating domain design expressions
Querying	- retrieving and juding states of common images regarding design developments manifested in domain design expressions	- reflecting on domain design developments in respect of what emerges as states of common images

Fig. 8 When situated in the settings of coupled group/individual modeling spaces, modeling acts become communicative acts that require participation and coordination among designers working in different modeling domains.

- *Mapping* -- the act of translation and integration of (parts) of an existing conceptual structure to (parts) of another conceptual structure.
- *Constructing* -- (given a conceptual structure) the act of applying operations onto selected primitives for creation/modification of *CI* or *DDEs*.
- *Querying* -- (given a conceptual structure) the act of applying operations onto parts of *DDEs* or *CI* for evaluations of design instances.

The Coupling of modeling spaces

Based on the previous observations of how common images and domain design expressions are formed and changed, it can be inferred that there are two distinctive ways of interconnecting a group modeling space with multiple individual ones.

- {(*CI* /GMS) → (*DDEs*/IMSs)}
  Creating, storing, and updating common images in a group modeling space leads to the creating, storing, and updating of domain design expressions in distributed individual modeling spaces. This coupling of group-individual spaces enables modeling acts to be performed directly on common images modeled in a group space which can consequently affect the states of local design expressions. The funicular modeling is an example of such an interconnection between group and individual modeling spaces.
- {(*DDEs*/IMSs) → (*CI* /GMS)}
  Creating, storing, and updating local design expressions in any individual modeling space leads to the creating, storing, and updating of the state of a common image. This coupling facilitates direct modeling acts performed in individual spaces which, consequently, can trigger the the projection of changed common images.in a group space. The coupling of the spaces for scoring, diagraming, and projecting in the fountain design is an example.

To conclude the current exposition of participatory design, the performing of modeling acts and the coupling of modeling spaces are put together into a matrix [Fig. 8]. The example modeling acts are said to be situated in two types of participatory setting for modeling design objects, and modeling acts become *communicative acts* that require, or lead to, communication among designers working in different domains of modeling.

3.4 Related work

The problem of what and how to develop communication or computer systems that can be supportive for people involved in collaborative design has become an active research area within the field of Computer-Supported Cooperative Work (CSCW). In a recent CSCW bibliographical survey, Greenberg [Gre91] suggested the keyword "shared workspaces" to cover an emerging research area. In this subarea of CSCW research, the understanding of "collaborative design" is one of the focus objects of observational studies, and some research prototypes of "shared drawing spaces" have been developed to support, mainly, group drawing activities.[4] In contrast, this paper proposes a view of design as modeling activities, by which more attention is drawn to the issues of participants with heterogeneous backgrounds and the interrelations between modeling by sharing and by

[4] For a survey of CSCW-oriented designs in shared drawing space with a particular interest in reviewing how the issues of supporting collaborative design work have been addressed by the research prototypes, see [Pen92a].

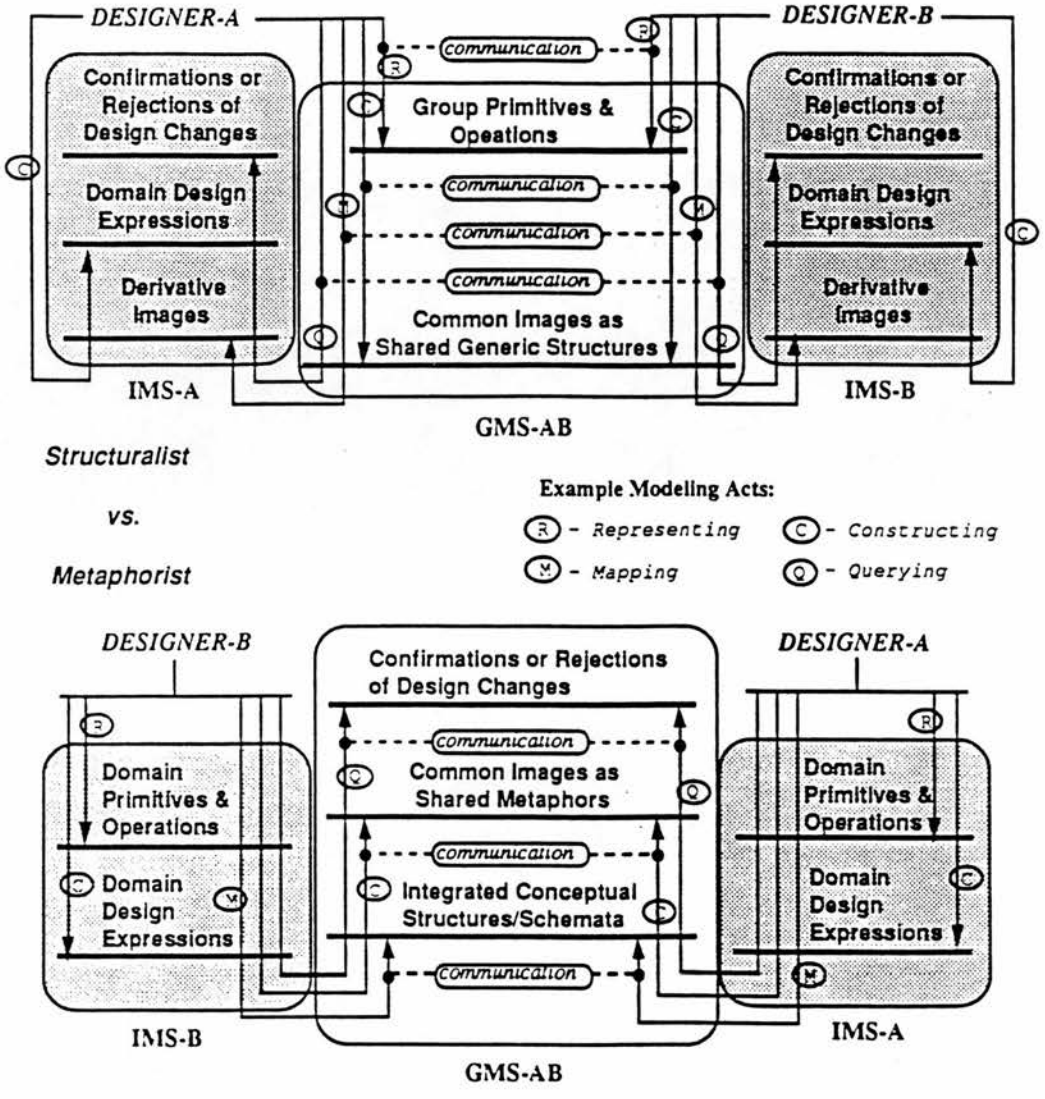


Fig. 9 Two abstract communication patterns found in the present study of participatory architectural modeling are characterized as *Structuralist* vs. *Metaphorist*. The number of designers indicated is arbitrary. The scaling of 2 to *n* designers can be envisaged by viewing this diagram as "a section of a cylindrical structure." Seen from this picture, in coordinating modeling activities with other members of a design team, an individual's workspace is a combination of his or her own IMS with a GMS.

Acknowledgements

The author thanks the joint guidance from John Lee and Aart Bijl in developing this paper. This research work was partially supported by the UK ORS Awards Scheme.

References

[AHMC85] C. Alexander, D. Howard, J. Martinez, and D. Corner. *The Production of Houses*. New York Oxford: Oxford University Press, 1985.

[Bon89] A. H. Bond. The cooperation of experts in engineering design. In L. Gasser and M. N. Huhns, editors, *Research Notes in Artificial Intelligence*, Vol. 2, pages 463--484. London: Pitman, 1989.

[Buc88] L. Bucciarelli. An ethnographic perspective on engineering design. *Design Studies*, 9(3):159--168, July 1988.

[CN83] G. R. Collins and J.B. Nonell. *The Designs and Drawings of Antonio Gaudi*, pages 31--35. Princeton, N. J. Guildford: Princeton University Press, 1983.

[Ers80] R. Erskine. *Byker Redevelopment: Byker Area of Newcastle upon Tyne,England, 1969-82*. Tokyo: A.D.A. Edita, 1980.

[Gre91] S. Greenberg. An annotated bibliography of Computer Supported Cooperative Work. *SIGCHI Bulletin*, 23(3):29--62, July 1991.

[Gru91] J. Grudin. Obstacles to user involvement in software product development, with implications to CSCW. In *International Journal of Man Machine Studies*, 34: 435--452, 1991.

[Hal69] L. Halprin. *The RSVP Cycles: Creative Processes in the Human Environment*, pages 54--57. George Braziller Inc., New York, 1969.

[HG88] N. J. Habraken and M.D. Gross. Concepts design games. *Design Studies*, 9(3):150--158, July 1988.

[Kro88] L. Kroll. *Lucien Kroll: Buildings and Projects*. London: Thames and Hudson, 1988. English translation by Joseph Masterson, introduction by Wolfgang Pehnt.

[Kuu91] K. Kuutti. The concept of activity as a basic unit of analysis for CSCW research. In L. Bannon, M. Robinson, and K. Schmidt, editors, *Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, pages 249--264, Kluwer Academic Publishers, 1991.

[Lam77] W. M. C. Lam. *Perception and Lighting as Formgivers for Architecture*, pages 125--129. McGraw-Hill Book Company, 1977.

[Law80] B. Lawson. *How Designers Think*, pages 171--186. The Architectural Press Ltd: London, 1980.

[Mar79] C. Martinell. *Gaudi: his Life, his Theories, his Work*, page 335. Barcelona Editorial Blume, 1979.

[Mid67] M. Middleton. *Group Practice in Design*, page 279. London: The Architectural Press, 1967.

[Pen92a] C. Peng. A survey of CSCW designs in shared drawing spaces. EdCAAD working paper, Department of Architecture, University of Edinburgh, June 1992. In submission.

[Pen92b] C. Peng. A formal analysis on teamwork in design modelling. *Edinburgh Architecture Research*, 19:137--154, 1992.

[SB91] K. Schmidt and L. Bannon. *CSCW, Or What's In A Name?* July 1991. Paper submitted for publication.

[Sta89] S. Leigh Star. The structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving. In L. Gasser and M. N. Huhns, editors, *Research Notes in Artificial Intelligence*, Vol. 2, pages 37--54. London: Pitman, 1989.

[Sch80] D.L. Schodek. *Structures*. London: Prentice-Hall, 1980.

[Sch85] D. Schon. *The Design Studio: An Exploration of its Traditions and Potentials*, pages 30--52. London: RIBA Publications Ltd., 1985.

[TL88] J. C. Tang and L.J. Leifer. A framework for understanding the workspace activity of design teams. In *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW '88)*, pages 26--28, ACM Press, 1988.

[War87] T. Ward. Design archetypes from group processes.*Design Studies*, 8(3):157--169, 1987.

A FORMAL PERSPECTIVE ON TEAMWORK IN DESIGN MODELLING

Chengzhi Peng

EdCAAD
Department of Architecture
University of Edinburgh

ABSTRACT

How individual members of a design team can use their own concepts and methods for producing design expressions, and how those expressions can be correlated and integrated with other expressions composed by other designers using different means is formally described and analysed. This paper presents a general framework for describing what task structures may characterise teamwork in design modelling. Taken as a key issue, the representation and use of spatial concepts by different individuals for collaborating on shape construction is investigated computationally. A formal system of modular algebraic specification is used to represent some example concepts which are then used to model the potential interactions between members of a design team. The current formal approach arrives at an indication that how designers communicate and coordinate can be better and more precisely characterised by looking at three aspects of teamwork in design modelling: (a) co-specification of spatial operations, (b) joint provision of source descriptions, and (c) coordination in making design changes.

KEYWORDS: spatial constructs, shape construction, spatial operations, teamwork, design modelling, OBJ3, modular algebraic specification.

1. INTRODUCTION

Studies and experiences in architectural design have shown that the forming and exercising of *spatial concepts* are basic to the practice of design modelling concerning, particularly, the construction of shapes or form [5] [8]. It is evident that designers working in different aspects of a building design, such as architectural layout, structure, interiors, lighting, landscaping or mechanical engineering etc, tend to formulate and act upon spatial concepts with different features, which make up the *modelling methods* that the individuals often work with. But there is also the fact that most buildings have been erected, more or less, in accordance with the joint intentions of the designers who, working together as design teams, designed the buildings. Given these observations, we are motivated to enquire into the problem -- "How do designers collaborate on shared design projects and achieve unity in buildings on the basis of individually oriented modelling methods?".

From a computer-aided design point of view, our investigation of the problem is aimed at a conceptual understanding of the nature of teamwork in design modelling. The basic understanding is essential in establishing explicitly the principles of designing *collaborative* computer-aided design environments which are capable of supporting the needs for communication and coordination among members of a design team in the course of constructing design models based on individual contributions.

The current study presents a working example that might come to terms with a *reflexive* perspective of Computer-Supported Cooperative Work (CSCW) that argues the importance of catering for the individual's requirements and preferences in collaborative environments [2] [13]. Teamwork in design modelling is another interesting but largely unexplored problem domain of research in CSCW. According to our investigation, teamwork in design modelling does have some features that are different from what have been reported in the designing of computer systems for supporting other kind of group activities (typically, of collaborative writing [12] [3]). Among many other points, if designing and implementing groupware for supporting collaborative design modelling is considered, it would be impractical to impose any pre-defined product structures or working procedures as common task environments that members of a design team have to work with. To meet the requirements for supporting teamwork in design would demand that we think about the matter in another way; that is to say, product structures are actually the outputs of using the groupware by a team of design participants, and a system can be more supportive if it is capable of mediating the working procedures or relationships that are developed by the collaborative users themselves in the course of designing. This seems to indicate that the core of a collaborative modelling environment should be in general a *system framework* for the users (the building designers) to create individual design constructs and models, which then allow team members to express and interact with each others' design intentions and knowledge in modelling a common image with an underlying shared structure, perhaps, unique to each design project at hand.

In the rest of the paper, our account of teamwork in design is organised in the following manner. A brief review of other researchers' work is presented in contrast to the method we are using in the paper. An informal description of a working example is presented in the third section to illustrate our basic view of what constitutes teamwork in design modelling in some detail. Section Four gives an explanation of how an algebraic specification language is used to represent and model the aspects of the group work in the computers, and some interactive situations are discussed with respect to the modelling results. Finally, a concluding remark summarises a conceptual structure of communication and coordination for design modelling seen from the formal perspective.

2. RELATED WORK AND OUR APPROACH

In studying the aspects of group or social processes of design, several other approaches have been attempted. One widely accepted is, perhaps, the metaphorical representation of design as *game*. Lawson has reviewed four design games (CONNECT, GAMBIT, INHABS, URBISM) that were specially designed to explore the group dynamics of game playing, which was said to simulate how designers work with others [9]. More recently, Bucciarelli employed an *ethnographical* view in his account of design as a social process consisting of constraining, naming, and decision discourse [1]. Habraken and Gross invented a computer program called *Concept design games* which was taken mainly as a research tool for studying the interactive processes engaged in by the designers as the game players. This multi-user computer game was programmed to record the players' chains of moves, and by replaying the recorded games, the complex interactions involved in the control distribution and territorial organization can be investigated in a subsequent analysis [7]. In contrast to the physical games, the computerized game seems to offer the players more abstract means for making the game moves.

In the study of group work in design modelling, a somewhat different approach is adopted here. First of all, we think that there is the necessity of representing individualities which serve the foundation of teamwork in design modelling. This paper proposes that different sets of spatial concepts or constructs can be used to mark the differences between individuals, i.e., the design participants. Shapes, produced by the individuals using their personal spatial constructs, can then motivate or evoke interactions among designers, which lead to the group process of interrelating individual shapes into common spatial structures shared by the team members. For this reason, a highly conceptual programming language OBJ3 is introduced as an experimental apparatus which allows a *modular algebraic specification* approach to specify spatial constructs and to generate instances of shapes. Our use of the formal system shows that mapping between spatial constructs can be developed as one of the potentially useful task structures for designing CSCW in design modelling.

Secondly, regarding computational design, we have a rationale for formal (algebraic) specification of spatial constructs, which says that specifications of spatial constructs can be thought of as producing *shape structures* as a theory, and design is the actual use of these structures in the construction of the intended shapes from the theory[1]. Methodologically, by achieving a computational representation of an overall process of joint shape constructions, we have demonstrated an analysis of how an individual's spatial concepts, when symbolically specified, function as computational devices for participation in group modelling of shapes.

In our formal approach to the problem, the scope of describing teamwork in design modelling has to be restricted to simple shape construction, for the purpose of clarity. Given the simplicity, a general framework for describing and analysing teamwork in design modelling is proposed as a composition of the following individual and group tasks:

Individual Tasks:

- defining (personal) spatial constructs as individual modelling methods;
- generating shape instances in one's modelling method;

Group Tasks:

- communicating how shape instances can be related with one another by developing common operations for transforming and combining source shapes into integrated shapes as shared spatial structures;
- coordinating in changing parts of source shapes to yield different states of the shared integrated shapes.

---

[1]Here we may make a comparison; in the model-theoretical approach to software development, formal specification is thought of as producing a theory which describes the computing system to be built, and design is thought of as finding the most appropriate model of the theory [14].

3. A WORKING EXAMPLE OF TEAMWORK IN DESIGN MODELLING

In this section, a simple example of joint shape construction is presented. As a hypothetical, but possibly illuminating example, this exercise is contrived to illustrate a view of group work in design modelling, focusing on the aspect of shape structure. It begins with a description of two individual domains of spatial concepts that represent two participating designers' modelling methods. A development of joint shape construction is then introduced on the basis of the existing concepts.

Consider a design exercise of modelling a two-dimensional shape *Envelope*, consisting of two aspects: (1) *Enclosure*, and (2) *Opening*. For the Enclosure part, one of the participants, say, Designer A, is responsible; and A creates the modelling method, Method-A, which captures a way of constructing and manipulating the elements of Enclosure. Another participant, say, Designer B, is responsible for the modelling of Opening using another method, Method-B. Teamwork in this exercise is to find a way of integrating Enclosure with Opening into Envelope, under the circumstances that Envelope is modelled on the basis of what Enclosure means for A, and what Opening means for B, and what Envelope jointly means for both A and B[2]. Given the development of a contrived example for illustrative purposes only, we now describe the spatial concepts used by A and B as follows.

3.1 A Shape Construction of Enclosure by Designer A

Designer A, as an expert in designing spatial enclosure, has the following spatial notions which are now intuitively described as Method-A, which is supposed to be used by A for the task of modelling the shapes of Enclosure:

> **Method-A** (see Figure 1): For an enclosure to exist, we consider that there is an initial point named as **Pivot**. A number of directed lines, **Stems**, pass through this point and are defined by the position of pivot and their angles. Up to two **Nodes** can be located on each stem according to their distances from the pivot. Any segment connecting two nodes defines an **Edge**. A connected chain of edges then forms a closed area of **Enclo**, an enclosure in a space. An enclosure has the attributes of length and area; the length of an enclosure is the total of all the lengths of its constituent edges, and the area of an enclosure is the total areas of its constituent triangles defined by its edges and the pivot.

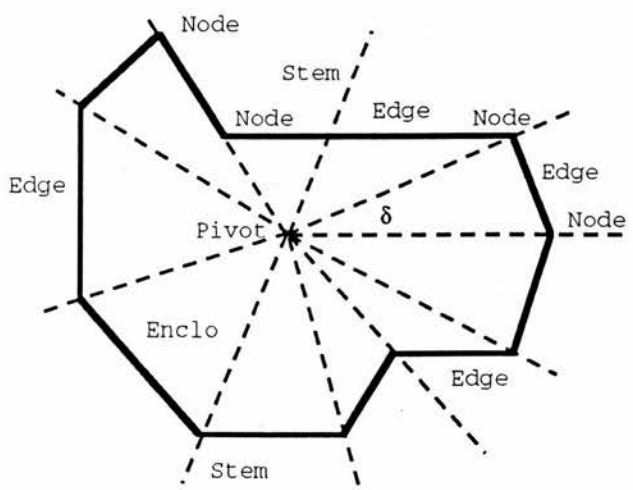


Figure 1: An instance of **Enclo** with an underlying shape structure described in Method-A by designer A.

3.2 A Shape Construction of Opening by Designer B

Suppose designer B, being experienced in window design, has particular concepts of what an opening would be with respect to the surrounding environment. We assume that Method-B is defined by B as follows:

> **Method-B** (see Figure 2): In deciding openings for a building, we start with a directed view line, **Vl**, which is decided by giving two initial reference points, view point, **Vp**, and view target, **Vt**. Two directed lines, left view line, **L-vl**, and right view line, **R-vl**, are defined by the position of the view point and two angles. Perpendicular to the view line, a screen line, **Sl**, can be located by giving a distance to the view point (**Vp**). An **Opening** is then decided by the intersections of the left view

[2]Rather informally, Enclosure can be thought of as referring to walls, or other elements, with the purpose of keeping the weather out. Opening can be thought of as windows, being elements added to allow light in and a view of the outside. Envelope can be thought of as the combination of Enclosure and Opening, encompassing the purposes of both.

line, and right view line with the screen line. We might thus name the two ends of an opening as α and β respectively; an opening has the attribute of length which is determined by the distance between its α and β ends.

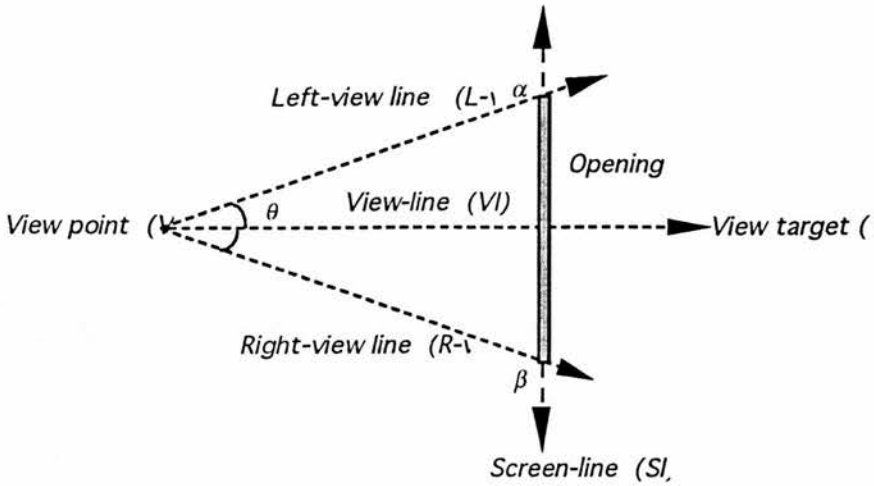


Figure 2: An instance of **Opening** with an underlying shape structure described in Method-B by designer B.

3.3 Participating in Modelling Envelope

The above descriptions of Method-A and Method-B introduce what individual modelling methods are to participate in the group design[3]. As shown, apart from their sharing some geometrical entities such as `point`, `line`, `directed-line`, and `distance-between-points` etc, these two methods have been individually defined without any connection between the constituent constructs represented in each method. Shapes of Enclosure or Opening can be modelled using the methods, possibly in different workspaces. Suppose that these individually modelled shapes are subsequently gathered by both designers in a common visual and cognitive space where A and B can display and view the instances of shapes. Naturally, it can be inferred that A and B shall set out discussions for how to relate the parts of the shapes in one way or another according to whatever they might think of[4].

Continuing to work on the example, we define that joint shape construction of an Envelope is participated in by A and B for developing the ways of combining the elements of **Enclo** and **Opening**. Conceptually, we use a descriptive schema of spatial *operations* to represent what might be evolved from the social process. The spatial operations thus co-created by the participants are applicable to the shape instances and yield integrated shapes representing the spatial structures shared by the designers as we proposed in Section Two. Computationally, there is a point to be made concerning the construction of common spatial structures. Note that in a spatial structure, the underlying structures of participant shapes defined in the individual methods remain as parts of the spatial structure generated by particular spatial operations; but it is possible to define new structural elements on top of the participant ones. To illustrate the point, two example spatial operations are given in the following as a partial representation of joint construction of Envelope by A and B.

The *On-cutting* operation is devised to transform (to cut) an Edge (of Method-A) with an imposed Opening (of Method-B) such that a new element, named Portion (a shared construction), can be generated with a structure of two emerging Edges and the given Opening. To see the structure, a more formal description of *On-cutting* is stated as follows:

> ***Spatial Operation 1*** [**On-cutting**] (see Figure 3) Given an Edge and an Opening, if the α end and the β end are *on* the Edge, then a Portion is constructed by (1) cutting the Edge with the Opening and yielding two Edges, (2) combining the two Edges of (1), with the Opening, into a Portion .

---

[3]Note that the two methods described here are not claimed to be standard or correct ones that are widely practised in designing building enclosure and opening, and we do not think that there exist such methods in practice or in theory. Even the splitting of envelope into enclosure and opening is an arbitrary one. They are devised by us, as design concepts in reality are developed by the designers themselves, to illustrate what kind of thing we mean by individual modelling methods.

[4]The social aspect of group design described here may have relevance to what Walter Gropius advocated as "participation in the creative process of the basic space-conception"; a process Gropius thought so essential to the result of final unity in the building. [see Middleton 1967, p279.]

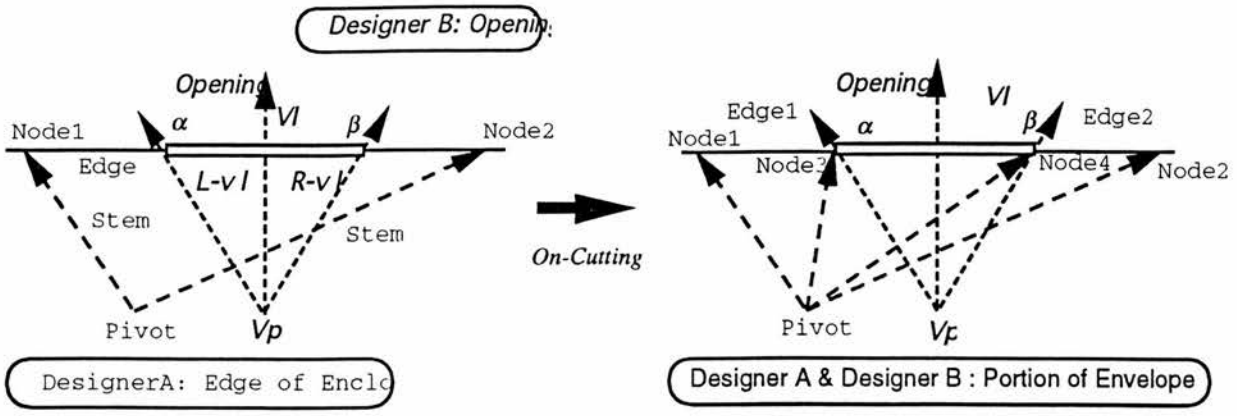


Figure 3: A sketch of making a shape of Portion out of an Edge and an Opening, if the Opening is *on* the Edge.

The second example is devised to deal with a different relative relation between an Edge and an Opening. For some particular reasons (getting a better view or more light, for instance), an Opening has to be displaced "outside" an Enclosure which may motivate A and/or B to define another way of cutting a local Edge and connecting the two Edges with the Opening. Again, the *Out-cutting* operation cannot be fully defined without the co-existence of the Edge and Opening structures and images. The following definition may show more clearly what interrelations are involved in *Out-cutting*:

Spatial Operation 2 **[Out-cutting]** (see Figure 4) Given an Edge and an Opening, if the Opening is *outside* the Enclosure, then a Portion is defined by connecting the Opening with the Edge in the following steps:

(i) Draw a line l_1 parallel to Vl (the View-line of the Opening) through the end of the Opening that is *farther from* the Edge, and get the intersection i_1 of l_1 and the Edge;

(ii) Draw a line l_2 perpendicular to the Edge through the end of the Opening that is *nearer to* the Edge, and get the intersection i_2 of l_2 and the Edge;

(iii) Draw l_1 through α and l_2 through β perpendicular to the Edge if the Opening is parallel to the Edge;

(iv) From left to right, make the four Edges: Edge1(Node1, i_1), Edge2(i_1, α), Edge3(β, i_2), Edge4(i_2, Node2);

(v) Portion is defined by: Portion(Edge1(Node1, i_1), Edge2(i_1,α),Opening(α, β),Edge3(β,i_2), Edge4(i_2, Node2)).

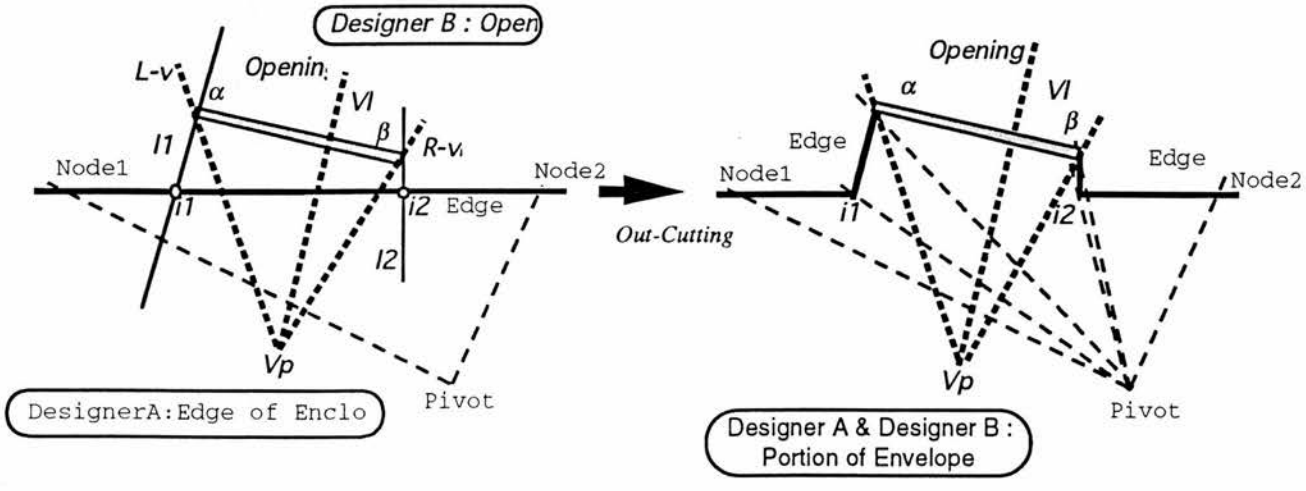


Figure 4: A sketch of making a Portion of Envelope out of an Edge of Enclo and an Opening, if the given Opening is *outside* the given Edge of Enclo.

There are other spatial operations for the spatial conditions such as an Opening *across*, or *inside* an Edge that might be jointly developed by the two designers, but they are omitted here for simplicity. The point is that spatial operations can always be extended and modified in the on-going processes of exploring the possibilities of relating the individuals' shape constructions. In our case, with the emerging concepts that are functional across the methods of Enclosure and Opening, a shared modelling method, say, Method-C (or Envelope method), may thus be created to

accommodate a number of spatial operations. Notably, it is Method-C that enables a joint construction of a **Portion** out of an **Edge** and an **Opening**. We may say that the functionalities of Method-C constitute a common workspace where the shape descriptions given by designers A and B can be transformed and combined into integrated shapes as shared spatial structures according to what set of *spatial conditions* is met.

According to the example worked so far, teamwork in design modelling can be said to take place at two different levels: (a) at a lower level, A and B jointly define a common base of spatial operations on the basis of their own modelling methods; (b) at an upper level, the two designers work together for the modelling of the integrated shapes as shared constructions by applying the spatial operations to the source shapes that are provided jointly in the individual methods. [For (b), there are more detailed explanations in Section 4.2.]

We have formally investigated the above working example of interrelating the individual methods into a common platform for joint shape construction. This investigation was done by formalising the example into a computational representation using the functional specification and programming language OBJ3 [11]. In the computational experiment, we demonstrate how spatial concepts, modelling methods, and spatial operations might be formally specified, represented, and used with respect to the aspects of teamwork illustrated in the example.

4. AN ALGEBRAIC SPECIFICATION FOR MODELLING JOINT SHAPE CONSTRUCTION

A full introduction to the formal basis of modular algebraic specification in OBJ3 can be found in the original authors' description of the language [6]. An excellent application of modular algebraic specification that is relevant to our current approach, but with more emphasis on the formal discipline of program specification, and less on design practice, can also be found in [4].

Briefly, in our formal specification of the teamwork case using OBJ3, a *Geometry* module is specified to provide an experimental set of basic *geometric constructs* such as POINT, ANGLE, LINE, CIRCLE, SEGMENT, ... etc, so that they can be imported in the specifications of the two individual systems[5]. Each modelling method has been specified as a modular system consisting of a set of interconnected OBJ3 *objects*, corresponding to the concepts described in the method. These two modular systems are developed independently of each other (both logically and functionally), representing the two separate design concerns. The spatial operations such as *OnCut*, *OutCut*, and *InCut*, which constitute the common operations for modelling the instances of Envelope, are then specified on top of the objects of the participating modular systems.

With the mathematics of *order sorted algebra* and the *modularization* mechanism supported by OBJ3, we firstly trace out a more detailed structure of the group tasks. Communication and collaboration in joint shape construction are necessarily involved in performing the following tasks:

(1) specification of spatial operations;
(2) provision of source shape descriptions; and
(3) modification and evaluation of integrated shapes.

4.1 Co-Specifying Spatial Operations

We find that the specification of spatial operations involves correlating the spatial concepts contributed by one member with those by another, leading to the domain-crossing functionalities for modelling integrated shapes. The task of specifying spatial operations, however, can only be achieved by collaboration among designers. The process of co-specification shows that:

(a) how spatial operations get named is dependent on the agreements among the authors such that they are commonly recognizable to all parties;
(b) no one designer could have complete and precise knowledge about what modules to import, and the declaration of shared objects also relies on a common recognition;
(c) the *form* of operations, i.e., the arrangement of sorts (types) of participating shapes together with the sorts of resultant shape, can only be declared on the basis of what shape constructs are accessible in the participants' modelling methods;
(d) the *behaviour* of a spatial operation, i.e., the set of axioms or theories as term rewriting rules and the conditions of applying these rules, can only be spelled out by consultation and collaboration.

An immediate question is what conditions such an interactive process.

---

[5]The collection of basic constructs, in the form of abstract data types, representing the set of primitive geometrical entities available for participants to use, is in turn built upon the OBJ3 system built-in data types such as `FLOAT` (for real numbers), `BOOL` (for Boolean expressions), `NAT` (for natural numbers), `INT` (for integers) etc.

The structure diagram in Figure 5 shows how an operation of reconstructing Node with α and Pivot might be co-specified by the two designers[6]. The data type Node (of Method-A) is reconstructed with the source data types of α (of Method-B) and the Pivot (of Method-A). As we can see, in the reconstructing of Node with α and Pivot, A, the person representing the knowledge of Node, is able to complete this specification if and only if B collaborates with A by providing the knowledge of mapping α to Point and to two Floats. With these mappings available, an expression of Node can now be computed given a pair of the α end of Opening and the Pivot.

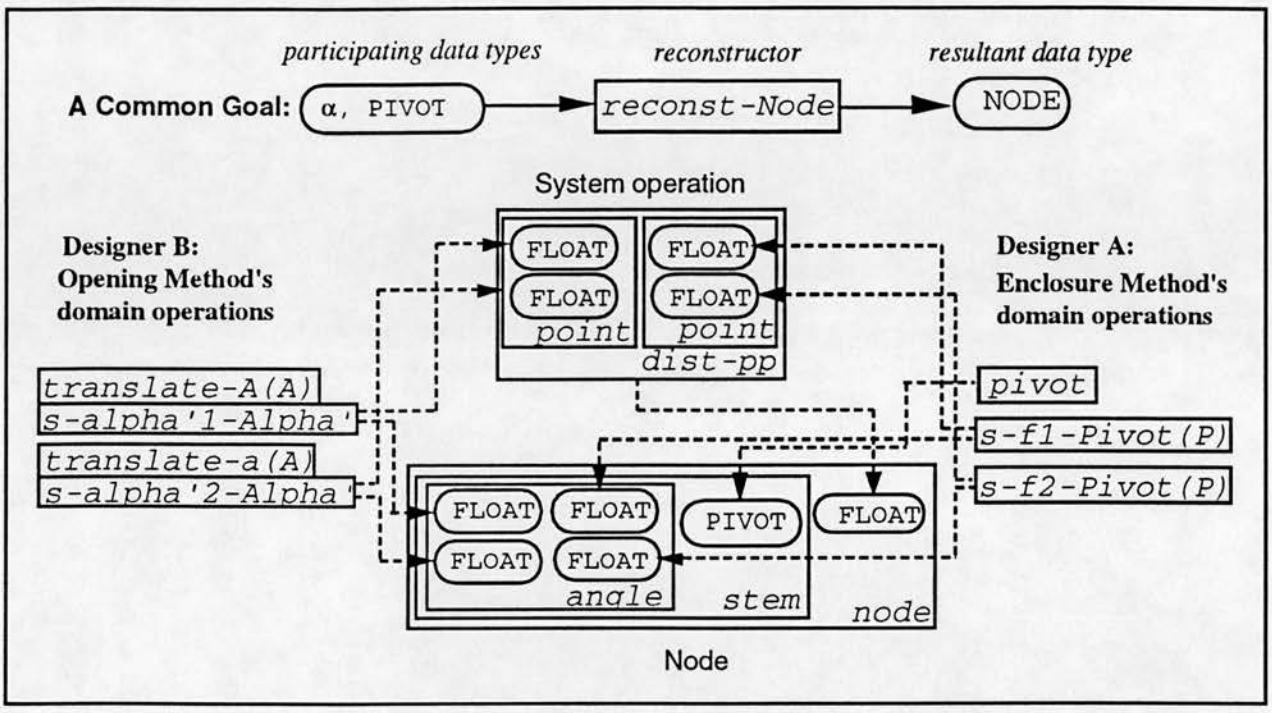


Figure 5: A structure diagram shows the collaboration for a joint specification of reconstructing Node with α and Pivot.

The above structure diagram suggests a picture of collaboration between A and B. Firstly, there is a *common task goal* specified by "α × Pivot → Node". To achieve the common goal, A and B share a *common cognitive map* made of the algebraic structure of Node and that of the system operation dist-pp which computes the distance between two points. The common cognitive map enables A and B to collaborate on "how" to achieve the common goal by constructing operations which translate the data types defined in their own modules back to the geometric data types defined in the geometry module. The "translators" constructed in individual domains thus become the functions for integrating the elements of spatial concepts originating from the designers. Although our example merely shows a portion of how common spatial operations may be co-specified by collaborative designers, the principle of specifying common goals and presenting cognitive maps is general enough to capture the semantics of cooperation in constructing complete spatial operations.

4.2 Joint Provision of Source Shapes

When a number of spatial operations are jointly specified as common operations for integrating domain concepts and instances, the *use* of the set of operations presents another issue of collaboration. Like the use of personal modelling methods in producing domain instances, the use of spatial operations leads to the production of integrated shapes; but the latter has to take place in a group process. The necessity of cooperation arises because the use of the spatial operations to generate integrated shapes is conditioned by providing the operations with *valid* source shapes. This implies that the source shapes need be provided jointly by the parties who have co-specified the operations. The goal for the designers is, therefore, to achieve the provision of valid source shapes. In the use of spatial operations for modelling parts of Envelope, at least four kinds of shapes can be differentiated conceptually among *participating shape*, *source shape*, *integrated shape*, and *resultant sub-shapes*.

- *Participating shapes* are the initial instances generated by designers' using individual modelling methods (such as Enclosure and Opening in our case);

---

[6]This is one of the two auxiliary operations that contribute to a full co-specification of *OnCut*.

- A *source shape* is the combination of participating shapes, which can be further subjected to the application of a certain spatial operation;
- *Integrated shapes* refer to the computational results of applying spatial operations to the source shapes;
- *Resultant sub-shapes* are the constituent shapes decomposed from an integrated shape.

Modelled in OBJ3, a source shape can be provided by simply "putting together" the participating instances, adhering to the algebraic structure governed by the spatial operations. More interestingly, source shapes can be constructed with "dependent" shape instances. It is revealed from our representing the events that the construction of dependence is open to the designers' individual or joint intention of what parts of participating instances to correlate and how they may be related.

As an example, Figure 6 shows an OBJ3 modelling of the joint provision of a source shape in which the `Opening` is constructed as being dependent on the `Edge` and an integrated shape of `Portion` is computed by the spatial operation of *OnCut*. In the source shape, the two participating shapes are related by designer B's displacing the `Screen-line` in relation to the intersection of the `View-line` and the `Edge`, such that the `Opening` is always *on* the `Edge`. For designer B to achieve this dependent construction in his describing the Opening part of the source shape, a `Segment` representation of `Edge` has to be made available by designer A.

As shown clearly in the OBJ3 expressions, to work with B, A has to provide the mapping from `Edge` to `Segment` which is the domain of A's knowledge. Without incorporating the mapping, B is unable to construct a valid Opening that is a dependent construction regarding A's Edge. Drawn from the example, all the geometric primitives can, in theory, be used to map parts of a participating shape in one domain to their corresponding geometric entities which can then be combined with other participating shapes into the description of a source shape. Furthermore, an integrated shape, with a different sort definition from the participating ones, can be decomposed into a collection of resultant sub-shapes which are recognisable and distributable to the individual modelling methods.

OBJ3 EXPRESSIONS — **EXPLANATION**

```
reduce mkPortion
(edge(node(stem(pivot(20,7),150),20),
      node(stem(pivot(20,7),30),20)),
```

} **A's participating shape of** `Edge`.

```
isOpen(isAlpha(sl(vl(vp(28,15),vt(28,70))
  ,distance-pp(point(28,15),intersect(
  segment(point(28,15),point(28,70)),
  segment(node-to-point(node(stem(pivot(
          20,7),150),20)),
          node-to-point(node(stem(pivot(
          20,7),30),20))))),
  l-vl(vp(28,15),150)),
      isBeta(sl(vl(vp(28,15),vt(28,70)),
  distance-pp(point(28,15),intersect(
  segment(point(28,15),vt(28,70)),
  segment(node-to-point(node(stem(pivot(
          20,7),150),20)),
          node-to-point(node(stem(pivot(
          20,7),30),20))))),
  r-vl(vp(28,15),30)))) .
```

} **B's participating shape of** `Opening`:
B's Opening is constructed dependently on the A's Edge by incorperating the mapping between Segment and Edge provided by A. (shown in the underlined expressions).

```
rewrites: 3123
result Portion:on-cut(
edge(node(stem(pivot(20.0,7.0),150.0),20.0),
     node(stem(pivot(20.0,7.0),65.60),10.98)),
```

Integrated shape of `Portion` **has parts:**

} **Resultant Subshape1 of** `Edge`;

```
mkOpen'(mkAlpah'(24.54,17.0),mkBeta'(31.46,17.0)),
```

} **Resultant Subshape2 of** `Opening`;

```
edge(node(stem(pivot(20.0,7.0),41.10),15,21),
     node(stem(pivot(20.0,7.0),30.0),20.0)))
```

} **Resultant Subshape3 of** `Edge`.

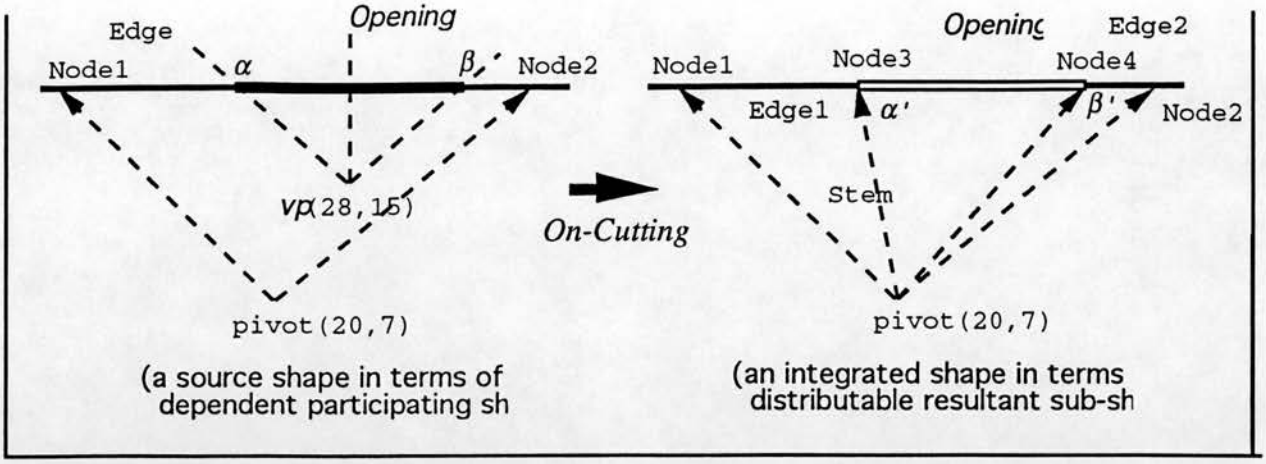


Figure 6: An OBJ3 modelling of the joint provision of a source shape consisting of dependent participating shapes, and the result of an integrated shape.

4.3 Coordinating in Making Design Changes

A natural consequence of constructing source shapes with dependent participating shapes is the coordination between designers in modifying the source shapes and thus the resultant integrated shapes. Why coordination is necessarily involved may be explained more plainly by the following general account:

Consider that two designers, say, A and B are engaged in a joint provision of source shapes. We denote the participating shape with the sort Ω designed by A as PS_Ω, and the one with the sort Ψ by B as $PS\psi$. A source shape can then be provided as $j(PS_\Omega, PS\psi)$ if the j operation is commonly used to combine the participating shapes. By executing $j(PS_\Omega, PS\psi)$, a resultant integrated shape with the sort Φ, is computed as IS_Φ which is shared by A and B.

In case $PS\psi$ is constructed dependently on PS_Ω by B, then changes made in PS_Ω by A will consequently alter the current states of $PS\psi$ as well as IS_Φ. More specifically, the alteration is to be reflected on the constituent shapes of IS_Φ, that is the resultant sub-shapes of RS_Ω, $RS\Psi'$, in $IS_\Phi = k(RS_\Omega, RS\Psi')$ where k is a predicate indicating what specific spatial operation has been performed by the system[7]. Note that RS_Ω, $RS\Psi'$ are recognisable to A and B respectively. Therefore, the above change of PS_Ω to PS_Ω' made by A shall result in $IS_\Phi' = k(RS_\Omega', RS\Psi')$, for which B may agree or disagree with $RS\Psi'$.

A juxtaposition of Figure 7.1 and Figure 7.2 shows an OBJ3 modelling of such a condition for coordinating in making design changes. Pictorially, the coordination occurs in the follow situation: Overlapping a shape of `Edge` given by designer A, designer B constructs a shape of `Opening` (such that the latter is dependent on the former); these two shapes are further combined by A or B to which a spatial operation, *OnCut*, is applied and produces the resultant integrated shape of `Portion` comprised of two resultant sub-shapes of `Edge` and one sub-shape of `Opening'`[8]; later on, A has a different intention of the participating `Edge` and makes changes in it. A's change leads to the generation of new state of the Portion shape, in which the shape of `Opening'` is altered, since the Opening shape got changed at the moment of A's changing the Edge shape. Assume there is a way to inform designer B of the changing state of the resultant `Opening`, which is his design responsibility, then such a computational consequence should trigger further communication and negotiation between A and B. It could be the case that the changes propagated to the shape of `Opening'` may not be intended by designer B; or more positively, B is able to suggest to A that further modifications can result in a better integrated shape of `Portion`.

---

[7]The variation of sort specification from $PS\psi$ (participating shape of sort Ψ) to $RS\psi'$ (resultant subshape of sort Ψ') is caused by the dependent construction in which the same design entity has varied algebraic structures for accommodating the mapping of some parts of PS_Ω to selected geometric entities.

[8]In OBJ3, it is specified that *mkPortion* : *Edge* × *Opening* → *Portion*, *oncut* : *Edge* × *Opening* → *Edge* × *Opening'* × *Edge*, where *Opening'* is another abstract data type representing an Opening entity.

```
reduce mkPortion(
edge(node(stem(pivot(20,7),150),20),
     node(stem(pivot(20,7),30),20),
isOpen(isAlpha(sl(vl(vp(22,12),vt(22,45)),
    distance-pp(point(22,12),
    intersect(segment(point(22,12),point(22,45)),
            segment(node-to-point(
              node(stem(pivot(20,7),150),20)),
                  node-to-point(
              node(stem(pivot(20,7),30),20)))))),
       l-vl(vp(22,12),150)),
       isBeta(sl(vl(vp(22,12),vt(22,45)),
       distance-pp(point(22,12),
       intersect(segment(point(22,12),point(22,45)),
                segment(node-to-point(
                 node(stem(pivot(20,7),150),20)),
                       node-to-point(
                 node(stem(pivot(20,7),30),20)))))),
      r-vl(vp(22,12),30)))) .
result Portion: on-cut(
edge(node(stem(pivot(20.0,7.0),150.0),20.0),
     node(stem(pivot(20.0,7.0),123.66),12.01)),
mkOpen'(mkAlpha'(13.34,17.0),mkBeta'(30.66,17.0)),
edge(node(stem(pivot(20.0,7.0),43.17),14.62),
     node(stem(pivot(20.0,7.0),30.0),20.0)))
```

Figure 7.1: An integrated shape of Portion results from an application of the mkPortion operation.

```
reduce mkPortion(
edge(node(stem(pivot(15,7),150),15),
     node(stem(pivot(15,7),30),15)),
isOpen(isAlpha(sl(vl(vp(22,12),vt(22,45)),
    distance-pp(point(22,12),
    intersect(segment(point(22,12),point(22,45)),
            segment(node-to-point(
              node(stem(pivot(15,7),150),15)),
                  node-to-point(
              node(stem(pivot(15,7),30),15)))))),
       l-vl(vp(22,12),150)),
       isBeta(sl(vl(vp(22,12),vt(22,45)),
       distance-pp(point(22,12),
       intersect(segment(point(22,12),point(22,45)),
                segment(node-to-point(
                 node(stem(pivot(15,7),150),15)),
                       node-to-point(
                 node(stem(pivot(15,7),30),15)))))),
      r-vl(vp(22,12),30)))) .
result Portion: on-cut(
edge(node(stem(pivot(15.0,7.0),150.0),15.0),
     node(stem(pivot(15.0,7.0),70.40),7.96)),
mkOpen'(mkAlpha'(17.67,14.5),mkBeta'(26.33,14.5)),
edge(node(stem(pivot(15.0,7.0),33.50),13.59),
     node(stem(pivot(15.0,7.0),30.0),15.0)))
```

Figure 7.2: A change made in the Edge shape results in a change to the source Opening shape, and then generates a new Portion shape in which the resultant sub-shape of Opening has been displaced and reduced in length.

5. CONCLUSIONS

In this paper, to describe the complexity of participation and interaction in a precise and coherent way, we have limited our coverage of teamwork in design modelling to focus closer examination on shape construction using a formal specification system. We have presented a formal description and analysis of teamwork in design modelling by examining a working example of joint shape construction in an algebraic specification language. From the formal perspective, we have identified at least three factors that condition collaboration among designers: firstly, the development of individual spatial concepts with which the shapes of interest in some domain can be constructed, manipulated, and evaluated; secondly, the development of joint spatial concepts in which the integrated shapes and the functionalities for modelling these shapes are co-specified; and thirdly, shared spatial structures can be generated by applying common spatial operations to the source shapes with valid structures.

Clearly, the current study indicates teamwork at two levels: Specification (**S**) and Instantiation (**I**). At the **S** level, Individual Modelling Methods (IMMs) are firstly built on a Geometry modular system whose specification in turn is based on the OBJ3 system built-in primitives. Each IMM consists of a set of interconnected spatial concepts that are formally specified as (initial) algebras. On the basis of participating IMMs, spatial operations can be jointly specified as a Shared Modelling Method (SMM). It is evident that a SMM is not simply a summation of multiple IMMs, but a functional network accessing and correlating the abstract data types distributed in the IMMs.

At the **I** level, instances of shapes are modelled either in IMMs or in a SMM. Participating shapes can be constructed, firstly, in each IMM, and then related by mappings and integrations. Source shapes provided in a valid form can be subjected to spatial operations according to what spatial conditions are met, yielding resultant integrated shapes. It is shown that modifications of parts of source shapes may very likely lead to controversies or disagreements, and thus motivate communication and coordination between the members of a design team.

ACKNOWLEDGEMENTS

The author would like to acknowledge the supervision and comments given by John Lee and Aart Bijl. Thanks also to Zhaohui Luo at LFCS, and to Dejuan Wang, Peter Szalapaj, Chris Tweed, Satoshi Imamura, and James Lothian at EdCAAD for their constructive discussions.

REFERENCES

[1] Bucciarelli, L. An ethnographic perspective on engineering design. *Design Studies*, 9(3):159-168, July 1988.

[2] Cockburn A.J.G. and Thimbleby H. A reflexive perspective of CSCW. *SIGCHI Bulletin*, 23(3):63-68, July, 1991.

[3] Ellis, C.A., Gibbs S.J., and Rein, G.L. Groupware: Some Issues and Experiences, *Communications of the ACM*, January 1991, Vol.34, No.1, pp 39-58.

[4] Goguen, J.A. Modular algebraic specification of some basic geometrical constructions. In Kapur D. and Mundy J.L., editors, *Geometric Reasoning*, pages 123-153, Elsevier Science, 1989.

[5] Goldschimit, G. Interpretation: its role in architectural design. *Design Studies*, 9(4):235-245, 1988.

[6] Goguen, J.A. and Winkler, T. *Introducing OBJ3*. Technical Report SRI-CLS-88-9, SRI, 1988.

[7] Habraken, N.J. and Gross, M.D. Concepts design games. *Design Studies*, 9(3):150-158, July 1988.

[8] Janke, R. *Architectural Models*. London Academy Editions, 1978.

[9] Lawson, B. *How Designers Think*. Butterworth Architecture, second edition, 1990.

[10] Middleton, M. *Group Practice in Design*, The Architectural Press, London, 1967.

[11] Peng, C. *A modular algebraic specification approach to shape construction.* 1991. Working paper, EdCAAD, University of Edinburgh.

[12] Selger, R. *Design and Implementation of a Distributed Program for Collaborative Editing*, MIT/LCS/TR-350, Laboratory for Computer Science, MIT, 1986.

[13] Thimbleby, H. Anderson, S. and Witten, I.H. Reflexive CSCW: supporting long-term personal work. *Interacting with Computers*, 2(3):330-336, December 1990.

[14] Woodcock, J. and Loomses M. *Software Engineering Mathematics*. Pitman, 1988.